Universidad Carlos III de Madrid
Escuela Politécnica Superior

Departamento de Tecnología Electrónica
Doctorado en Ingeniería Eléctrica, Electrónica y Automática

Tesis Doctoral

HARDWARE DESIGN OF CRYPTOGRAPHIC ALGORITHMS FOR LOW-COST RFID TAGS

Honorio Martín González

Dirigida por

Enrique San Millán Heredia

Leganés, 2015

E-mail: hmartin@ing.uc3m.es

Address:

Grupo de Diseño Microelectrónico y Aplicaciones
Departamento de Tecnología Electrónica
Universidad Carlos III de Madrid
Av. de la Universidad, 22
Leganés 28918 — Spain

Autor: Honorio Martín González
Directores: Enrique San Millán Heredia

Firma del Tribunal Calificador:

	Nombre y Apellidos	Firma
Presidente: D^a.	Teresa Riesgo Alcaide	.....
Vocal: D.	Giorgio Di Natale	.....
Secretario: D.	Emilio Olías Ruíz	.....

Calificación: .....

Leganés, 27 de Enero de 2015.

Contents

List of Figures

List of Tables

List of acronyms

ASIC *Application-Specific Integrated Circuit*

ASK *Amplitude-Shift Keying*

C1G2: *Class-1 Generation-2*

DoS: *Denial of Service*

EEPROM: *Electrically Erasable Programmable Read-Only Memory*

EPC: *Electronic Product Code*

FPGA: *Field-Programmable Gate Array*

GE: *Gate Equivalents*

HF: *High frequency*

ISO: *International Organization for Standardization*

LCG: *Linear Congruential Generator*

LF: *Low Frequency*

LFSR: *Linear Feedback Shift Register*

NIST: *National Institute for Standards and Technology*

PLL: *Phase-Locked Loop*

PRNG: *Pseudo-Random Number Generator*

PSK *Phase-Shift Keying*

RFID: *Radio Frequency Identification*

RN16: *16-bit Random Number Generator*

RO: *Ring Oscillator*

STR: *Self-Timed Ring*

TRNG: *True-Random Number Generator*

UHF: *Ultra High frequency*

Abstract

Radio Frequency Identification (RFID) is a wireless technology for automatic identification that has experienced a notable growth in the last years. RFID is an important part of the new trend named Internet of Things (IoT), which describes a near future where all the objects are connected to the Internet and can interact between them. The massive deployment of RFID technology depends on device costs and dependability. In order to make these systems dependable, security needs to be added to RFID implementations, as RF communications can be accessed by an attacker who could extract or manipulate private information from the objects. On the other hand, reduced costs usually imply resource-constrained environments.

Due to these resource limitations necessary to low-cost implementations, typical cryptographic primitives cannot be used to secure low-cost RFID systems. A new concept emerged due to this necessity, *Lightweight Cryptography*. This term was used for the first time in 2003 by Vajda et al. and research on this topic has been done widely in the last decade. Several proposals oriented to low-cost RFID systems have been reported in the literature. Many of these proposals do not tackle in a realistic way the multiple restrictions required by the technology or the specifications imposed by the different standards that have arose for these technologies. The objective of this thesis is to contribute in the field of lightweight cryptography oriented to low-cost RFID tags from the microelectronics point of view.

First, a study about the implementation of lightweight cryptographic primitives is presented . Specifically, the area used in the implementation, which is one of the most important requirements of the technology as it is directly related to the cost. After this analysis, a footprint area estimator of lightweight algorithms has been developed. This estimator calculates an upper-bound of the area used in the implementation. This estimator will help in making some choices at the algorithmic level, even for designers without hardware design skills.

Second, two pseudo-random number generators have been proposed. Pseudo-random number generators are essential cryptographic blocks in RFID systems.

According to the most extended RFID standard, EPC Class-1 Gen-2, it is mandatory to include a generator in RFID tags. Several architectures for the two proposed generators have been presented in this thesis and they have been integrated in two authentication protocols, and the main metrics (area, throughput and power consumption) have been analysed.

Finally, the topic of True Random Number Generators is studied. These generators are also very important in secure RFID, and are currently a trending research line. A novel generator, presented by Cherkaoui et al., has been evaluated under different attack scenarios. A new true random number generator based on *coherent sampling* and suitable for low-cost RFID systems has been proposed.

Abstract

La tecnología de Identificación por Radio Frecuencia, más conocida por sus siglas en inglés RFID, se ha convertido en una de las tecnologías de auto-identificación más importantes dentro de la nueva corriente de identificación conocida como Internet de las Cosas (IoT). Esta nueva tendencia describe un futuro donde todos los objetos están conectados a internet y son capaces de identificarse ante otros objetos. La implantación masiva de los sistemas RFID está hoy en día limitada por el coste de los dispositivos y la fiabilidad. Para que este tipo de sistemas sea fiable, es necesario añadir seguridad a las implementaciones RFID, ya que las comunicaciones por radio frecuencia pueden ser fácilmente atacadas y la información sobre objetos comprometida. Por otro lado, para que todos los objetos estén conectados es necesario que el coste de la tecnología de identificación sea muy reducido, lo que significa una gran limitación de recursos en diferentes ámbitos.

Dada la limitación de recursos necesaria en implementaciones de bajo coste, las primitivas criptográficas típicas no pueden ser usadas para dotar de seguridad a un sistema RFID de bajo coste. El concepto de primitiva criptográfica ligera fue introducido por primera vez 2003 por Vajda et al. y ha sido desarrollado ampliamente en los últimos años, dando como resultados una serie de algoritmos criptográficos ligeros adecuados para su uso en tecnología RFID de bajo coste. El principal problema de muchos de los algoritmos presentados es que no abordan de forma realista las múltiples limitaciones de la tecnología. El objetivo de esta tesis es el de contribuir en el campo de la criptografía ligera orientada a etiquetas RFID de bajo coste desde el punto de vista de la microelectrónica.

En primer lugar se presenta un estudio de la implementación de las primitivas criptográficas ligeras más utilizadas, concretamente analizando el área ocupado por dichas primitivas, ya que es uno de los parámetros críticos considerados a la hora de incluir dichas primitivas criptográficas en los dispositivos RFID de bajo coste. Tras el análisis de estas primitivas se ha desarrollado un estimador de área para algoritmos criptográficos ultraligeros que trata de dar una cota superior del área total ocupada

por el algoritmo (incluyendo registros y lógica de control). Este estimador permite al diseñador, en etapas tempranas del diseño y sin tener ningún conocimiento sobre implementaciones, saber si el algoritmo está dentro de los límites de área impuestos por la tecnología RFID.

También se proponen 2 generadores de números pseudo-aleatorios. Estos generadores son uno de los bloques criptográficos más importantes en un sistema RFID. El estándar RFID más extendido entre la industria, EPC Class-1 Gen-2, establece el uso obligatorio de dicho tipo de generadores en las etiquetas RFID. Los generadores propuestos han sido implementados e integrados en 2 protocolos de comunicación orientados a RFID, obteniendo buenos resultados en las principales características del sistema.

Por último, se ha estudiado el tema de los generadores de números aleatorios. Este tipo de generadores son frecuentemente usados en seguridad RFID. Actualmente esta linea de investigación es muy popular. En esta tesis, se ha evaluado la seguridad de un novedoso TRNG, presentado por Cherkaoui et al., frente ataques típicos considerados en la literatura. Además, se ha presentado un nuevo TRNG de bajo coste basado en la técnica de muestreo por pares.

Introduction

Introduction

Auto-identification technologies have been used massively during the last 50 years and have become indispensable in our lives. Among these identification technologies Bar-code has been the most used one in the last three decades and it is almost present in all day-to-day products. Another promising identification technology is Radio Frequency Identification (RFID). Although RFID appears to be relatively new, it has been used in the field since the early forties with military purposes.

Along many years of research and development, RFID has reached a development degree where the cost and miniaturization make it feasible to be used with commercial purposes. The cost was the main drawback in the early stages of the development, but nowadays the miniaturization and the manufacturing process automation allow affordable costs and make this technology accessible to all kind of companies. In addition, the increasing standardization of RFID technology, carried out by ISO and EPCglobal, has allowed the deployment of RFID technology around the world. As a maximum exponent of standardization stands out EPC Class-1 Gen-2 (EPC C1G2 [47]), extensively used in retail industry.

RFID technology offers several advantages over bar-code, as for example the univocal identification of different items without the need of visual contact. This unambiguous identification also makes possible to distinguish different objects from the same family product. RFID systems can store additional useful data to the company or the user, for example, locations (time and space).

As sensitive information related to each item is stored into each tag, it is necessary to add security to RFID systems. A major difficulty in providing RFID tags with security functions comes from the scarcity of computational resources available in such platforms ([140, 125, 13]). For that reason, modern cryptographic solutions

based on difficult mathematical challenges that involve intensive computational operations are not suitable for this resource-constrained devices.

A new area in the field of cryptography has been developed in the last years. In 2003 Vajda et al. introduced the new concept of *lightweight cryptography*. Since then, this research line has become more and more important. Lightweight cryptography tries to give answer to the necessity of security in resource-constrained environments where computational resources are very limited.

Several contributions related to lightweight cryptography have been reported in the literature. A large number of these contributions present a very theoretical approximation but do not show a realistic approximation to the requirements imposed by the technology.

Motivation

The motivation of this thesis is to contribute in the field of lightweight cryptography oriented to low-cost RFID tags from the microelectronics point of view.

Despite the numerous contributions reported in the literature about lightweight cryptography, there is a lack of proposals that tackle in a realistic way the multiple requirements imposed by the technology.

- There are some very theoretical contributions that do not provide any proof of their lightweightness [122] [110].
- In many cases, arguments in favor of their lightweightness are based on the use of some operations that are generally considered inexpensive by the authors. However, these estimations are not always correct, and the implementation of some of these proposals greatly exceeds the area limit of 4K Gate Equivalents (GE is the normalization commonly used for these applications) [73][97][34].
- In other cases, the design turns out to be not so lightweight because of factors such as the bit length of the variables, the need for additional memory blocks –which is usually missed in the analysis of resources–, and the overhead imposed by selection and control logic. These and other aspects often make the final gate count much higher than expected [34][110].

In addition to the restrictions imposed by the technology, the standards add additional requirements. In the field of low-cost passive RFID, the EPC C1G2 standard is widely used in the industry. This standard establishes the physical and logical requirements for UHF (Ultra High Frequency) low-cost RFID systems. Among these requirements, there are demands concerning the security. Specifically, it is necessary to embed a pseudo-random/random number generator (RN16) in each tag in order to secure the communications with readers. These kind of generators are widely used in typical cryptographic applications.

Pseudo-random number generators (PRNGs) are deterministic algorithms that generate an unbiased *random* output. There are several proposals oriented to low-cost RFID proposed in the literature [105], [94], [27]. In these contributions area and throughput estimations are usually shown, but in general no data are provided about hardware architectures or power consumption.

True random number generators (TRNGs), are generators that use some physical processes to generate random numbers. TRNGs are typically implemented on FPGAs due to their flexibility and cost. There are several proposals presented in the literature [127],[147], [54]. The main weaknesses of these works are the vulnerability against some attack scenarios ([14],[90]) and the lack of portability.

The aforementioned issues motivate different goals to advance in the field of lightweight cryptography for low-cost RFID tags.

Objectives

The main objective of this thesis is to contribute in the research field of lightweight cryptography from the microelectronics point of view. In particular, our goal is to design and implement lightweight cryptographic algorithms devoted to low-cost RFID tags that comply with EPC Class-1 Gen-2 standard. The following specific objectives are aimed:

1. Studying the main algorithms and functions, as well as their implementations, used in lightweight cryptography, analysing their footprint area and suitability to be used in low-cost RFID tags.

2. As the EPC standard establishes the necessity of an embedded Random number generator (RN16):

2.1. Designing and implementing a pseudo-random number generator compliant with the EPC-C1G2 standard, obtaining the main metrics (area, power consumption and throughput).

2.2. Designing and implementing a true-random number generator, evaluating its suitability for low-cost RFID tags.

Based on these general objectives, we establish some more specific goals.

Regarding the first objective, it is intended to identify and analyse the main elements used in lightweight cryptographic algorithms. This analysis will be focused on the footprint area due to the fact that area and tag cost are closely related. In addition, taking into account the problem presented in the literature with the area estimation (the control logic is not included, memory blocks are not contemplated, etc.), it is planned to establish general guidelines to give an upper-bound of the footprint area algorithm. These guidelines are intended to be very useful for making some choices at the algorithmic level, even for designers without hardware design skills.

Concerning the second objective, the main standard used in the industry (EPC-C1G2) establishes the necessity of a random generator (RN16) in each RFID tag.

Specifically, for goal 2.1. regarding PRNGs, it is intended to study of the state of the art of PRNGs for low-cost RFID tags. Afterwards, the main aim will be to design and implement a pseudo-random number generator suitable for low-cost RFID tags and obtain experimentally some of the main metrics related to the constrains imposed by the technology.

With reference to the 2.2. objective, it is planned to study the state of the art of lightweight true-random number generators. In this study the main techniques used in TRNG design and the typically contemplated evaluation scenarios will be evaluated. Finally, the main goal will be to propose a lightweight TRNG suitable for low-cost RFID tags.

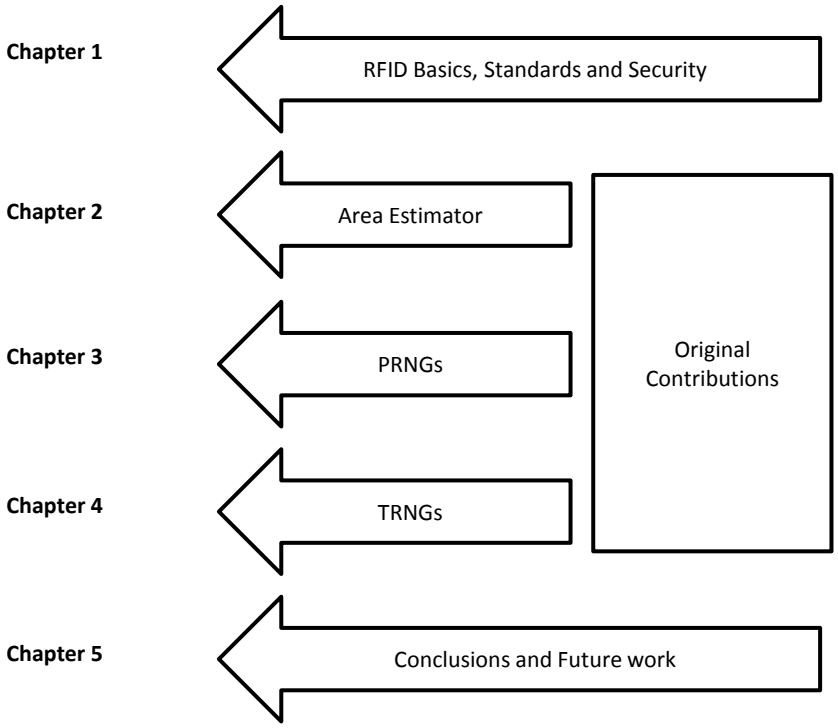


Figure 1: Thesis organization

Document Organization

As depicted in Fig.1, this thesis consists of five chapters organized as follows:

Chapter 1 shows an overview about RFID systems. The overview includes an explanation of the RFID basics (components of the RFID systems, main features, applications, etc), the most used standards (especial attention is given to EPC-C1G2 standard) and finally some remarks about RFID security.

In **Chapter 2** a study about the footprint area of the main lightweight cryptographic primitives is presented. Taking into account this information, a method to estimate the footprint area of a whole algorithm is proposed. Finally, our experimental results with a battery of real-world examples are discussed.

Chapter 3 presents the design and implementation of two pseudo-random number generators. Several architectures are proposed to improve some of the main metrics. Finally, both PRNGs are integrated into two authentication protocols. Resource experimental results of the complete security system are shown.

In **Chapter 4** a complete study of TRNGs suitable for digital devices is reported. After that a novel generator presented by Cherkaoui et al. is analysed, testing its response in typical evaluation scenarios of TRNGs (including two new attacks scenarios, clock and power glitches, not contemplated in typical evaluation scenarios). Finally, a new lightweight TRNG design based on the coherent sampling technique is proposed.

Chapter 5 summarizes the conclusions of this thesis and future lines are presented.

1

RFID Technology

As aforementioned, the abbreviation RFID stands for radio frequency identification. When the price of tags becomes economical enough, it is expected that RFID technology will increase efficiency in the field of real-time identification.

According to [40], the origins of this technology go back to the early 20's, when MIT developed the first prototype of RFID system. Afterwards, it was massively used with military purposes in the World War II. A plane identification system called *IFF* (Identification Friend or Foe) was the first time that RFID was used in the field.

In spite of the fact that RFID is an *old* technology, it is in recent time that companies have begun to understand its many benefits both for production and distribution. For example, the US Department of Defense and Wal-Mart require all their major suppliers to use RFID technology in their supply chains [67].

Nowadays, the Auto-ID labs are responsible of the development and deployment of this technology. The Auto-ID Labs are the leading global research network of academic laboratories in the field of Internet of Things (IoT). IoT is a hybrid network of the Internet and resource-constrained networks, including RFID. Auto-ID Labs and GS1 are the responsible of develop the Electronic Product Code Standard (EPC). This standard is the most widely used in the industry.

In this chapter, an overview about RFID technology is presented. It is structured as follows: In section 1.1 the basics of RFID systems are presented. First of all, the RFID operation and components are presented. After that, the main features of RFID systems are depicted. Finally, advantages, disadvantages and applications are shown. Section 1.2 introduces the RFID standardization focusing on EPC standard. Finally, Section 1.3 summarizes the security requirements of RFID technology.

1.1 RFID basics

1.1.1 RFID operation

A RFID communication system is based on a bidirectional communication between a reader (interrogator) and a tag (transponder) using radio-frequency waves. Typically, a reader sends a query to a tag population, obtaining a response with the unique identification number of each tag. This identification number is transmitted to the database, where it is associated to the corresponding information.

1.1.2 RFID components

RFID systems consist of three main elements: the RFID tag, the reader and the back-end database.

- **RFID tag**
  The RFID tags or transponders are attached to items to be identified. The information that will be transmitted in the communication process is stored in the tags. These days, common tags consist of a integrated circuit (generally a micro-processor) and a memory (see Figure 1.1). Other kind of tags known as *chipless* do not include an IC. These kind of tags are more effective in applications that implement simple functions. In addition, these tags are cheaper than the tags that contain an IC [89].

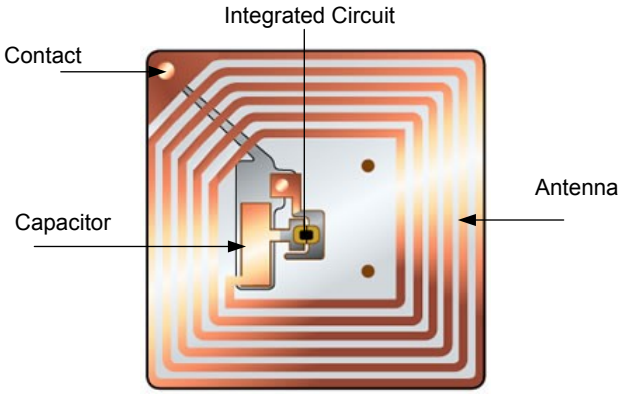


Figure 1.1: RFID tag [99]

It is possible to classify RFID tags attending to several features like cost, shape, material composition, etc [63]. But the two most extended classifications take into account the type of memory and the power supply source.

- **Type of memory:**
  - Read-only: This kind of tag has an ID that is usually established during the fabrication process and cannot be changed.
  - Write-once read-many: These tags do not have an ID after the manufacture process. The user establishes the ID but it can not be modified.
  - Fully rewritable: It is possible to change the ID of these tags multiple times.
- **Power supply source:**
  - Passive RFID tags: this kind of tag does not have embedded a power supply source. They rely on RF energy transferred from the reader to the tag. RF energy is used to power the tag and with communication purposes. This kind of powering has some drawbacks, for example the quality of the power signal or the range. On the other hand, this kind of tag uses a reduced area, that generally means a reduced cost. Typically, passive tags have a cost between 0.05 and 0.1 €.
    Other important feature of these tags is the frequency. Generally, the operation frequency is between 125-134 KHz or in the band of 13.56 MHz, although some tags reach up to 2.45 GHz [52]. The tag construction format is also an important feature and is usually selected taking into account the application and the work environment.
  - Semi-passive RFID tags: contrary to passive tags, semi-passive tags include a little battery. The IC of these tags is always powered. For that reason, the antenna is not optimized to collect energy from the reader. They are often oriented to the data transmission process.
  - Active RFID tags: this type of tag has an embedded power supply. Therefore, they offer a wide range of frequencies (55 MHz, 2.45 or 5.8 GHz) or distances (up to 100 meters). Their size is bigger than the previous ones. Typically they include an extra-memory. In addition,

this sort of tag, on the contrary than the previous one, can take the initiative in the communication process.

- **RFID Reader**

  The reader, also known as interrogator, is a device that collects and processes the data transmitted from tags. Depending on the application, some readers have the capabilities of writing in tags. When RFID systems involve the use of passive or semi-passive tags, readers play a key-role because they are responsible of powering the tags. In addition, readers usually carry out complex cryptographic operations. Moreover, they are the responsible of the communication with the back-end database.

- **Back-end database**

  The information stored in the tags is very limited. Usually, tags only store an index or an identification number (ID). In the back-end database this ID is related to the item information. Depending on the complexity of the RFID system, the back-end database can be omitted. A secure communication between the reader and the database is often assumed. As the resources are not constrained, they usually implement solutions like SSL/TLS [103].

1.1.3 RFID features

In this section some important RFID system features are presented :

- **Anti-collision:** It is possible to define two kind of collisions: multiple tags answering one reader or two readers interrogating the same tag. The first collision can be solved using probabilistic or deterministic anti-collision approaches. These anti-collision methods have been widely used in networks and have been redefined to be used in RFID systems. The second kind of collision, readers collision, can occur when the signal from one reader interferes with the signal from another reader where coverage overlaps. These collisions can be avoided using a Time Division Multiple Access (TDMA) protocol. TDMA establishes that readers must read at different times and do not interfere each other. The

problem is that a tag contained in the overlapping zone will be read twice. It is possible to solve this problem if the back-end database allows the tags to be read only once.

- **Coding:** Data coding is necessary to transmit digital signal through different channels. The secret of the air interface is that a reader has a very specific way in which it encodes data by modulation. The tag can not communicate with the reader without knowing how the information from the reader is encoded. The main coding procedures are: NRZ, Manchester, UnipolarRZ, DBP, Miller and differential coding on PP coding.

- **Modulation:** Generally, modulation is the process of varying one or more properties of the carrier signal, with a modulating signal that typically contains information to be transmitted. In this case, different modulating techniques are in some cases used in reader-tags communication and vice versa. The digital modulations typically used in RFID systems are: Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK) and Phase Shift Keying (PSK). They will be selected by taking into account power consumption, reliability and bandwidth requirements.

- **Energy transmission:** When the RFID systems consist of passive tags, it is necessary to establish a power transfer method between reader and tags. The most well-known techniques are:

  – Inductive Coupling: This technique is based on the magnetic coupling between the reader and tags. It works similarly to an electrical transformer. The reader's antenna generates a magnetic field which induces current in the tag's antenna. Tag's antenna consists of a coil and a capacitor. The induced current will charge the capacitor that provides the power to carry out the transmission.

    Systems using this technique must work in a near field (around the antenna's diameter) because the generated field strength will decrease quickly with the distance. In addition, the tag orientation will have an important effect in the transmission.

    Inductive coupling is generally used for Low Frequencies (LF and HF) due to the small coverage area. It is noteworthy the sensitivity of this technique

to electro-magnetic interferences. Moreover, energy can penetrate through non-conductive materials easily.

Typical applications include 1-bit inductive electronic article surveillance (EAS), anti-robbery systems, animal identification and access control.

- Backscatter: This technique is based on the propagation of electromagnetic waves. The reader transmits the energy using electromagnetic waves. The tags in the coverage zone receive part of this energy. The energy available is related to the distance to the reader's antenna. More precisely, it is proportional to the inverse of the squared distance ($1/d^2$).

  This kind of propagation is commonly used in high frequencies (UHF and microwaves). Finally, backscatter offers a wide coverage range, between 2 and 15 meters, but an embedded battery in the tag (active tags) is usually required. Due to the wide range, it is necessary a standard to define the spectrum range in this zone.

- Close coupling: This transmission technique is used in systems with a range between 0.1 to 1 cm. The tag is located in the middle of the reader's coil. It is the same operation way as the inductive coupling. This method is characterised by its high transmission efficiency. It is often used with tags that embed a powerful IC that consumes a lot of energy.

- **Frequency range:** Defining the operation frequency is one of the important aspects in the connection between readers and tags. The operation frequencies are selected depending on the application and the current standard.

  - Low Frequency (LF): This band includes frequencies from 0 to 135 KHz. One of the biggest advantages with LF is that it is not affected as much by surrounding metal. And the main drawbacks are the antenna cost and a slow data transfer.

  - High Frequency (HF):13,56 MHz is the frequency used for High Frequency RFID systems. This frequency is a global standard accepted and can be used world-wide. The advantages of this frequency are the tag cost and the transmission rate. On the other hand, surrounding metals may interfere with the RFID system.

  - Ultra High Frequency (UHF) and microwaves: The band corresponding to

868 MHz up to 2.45 GHz is used in UHF RFID systems. This frequency is the most common in the industries because it offers a wide read range. Moreover, it is standardized by the EPC Global. On the contrary, UHF RFID systems are affected by liquids and surrounding metals.

Table 1.1 summarizes the frequencies used in RFID systems, their main features and applications.

Frequency	Main Characteristics	Typical applications
Low Frequency (LF) Less than 135 KHz	- Widely used since 1980 - Tolerance to surrounding metals and liquids - Slow data transfer - little read range	- Animal identification - Industrial automation - Access control
High Frequency (HF) 13,56 MHz	- Extensively used since 1995 - Global standards - Bigger read range - Lower cost than LF tags. - Poor behaviour around metals	- pay cards - Access control - Anti-Counterfeiting - People monitoring
Ultra High Frequency (UHF) 860 up to 930 MHz	- Used since late 90's - Longer read range than HF systems - Low-cost Tags - Different standards depending on the region - Do not work in presence of liquids and metals	- Inventory control - Supply chain market - Active tracking
Microwave (SHF) 2,45 GHz and 5,8GHz -	- Used for several decades - Highest transfer rate - Commonly used with active and semi-passive tag - Read range similar to UHF systems - Affected by liquids and surrounding metals	- Access control - Electronic tolling - Industrial automation.

Table 1.1: RFID frequency range [52]

1.1.4 Advantages, disadvantages and applications

RFID technology offers numerous advantages against other identification systems. However, there are some drawbacks that have to be taken into account.

- Advantages: RFID systems are generally compared with bar-code systems. The main advantage of RFID tags is that they do not require a line of sight to be read. That feature makes possible to read concurrently several tags at the same time. The information about the item that provides the RFID system is superior to the information offered by the bar-code systems. In addition, RFID systems can distinguish unambiguously each item while bar-codes distinguish a family product. Since each tag can be unique, they can act as a security feature if lost or stolen. Other important benefit of some RFID tags is that they can be written several times.

- Disadvantages: In comparison with bar-codes, RFID tags are very expensive. Not only tag costs are important but also all the infrastructure necessary for integrating RFID technology into existing inventory control systems. Moreover, there are not internationally agreed frequencies for RFID operation, which implies that companies operating around the world have to take into account the regularization of different countries. Other important drawback is that it is difficult for an RFID reader to read the information in case of RFID tags installed in liquids and metal products. Finally, privacy concerns related to the lack of security have been present since the beginning.

Taking into consideration the different advantages and disadvantages presented above, RFID systems have been widely implemented for different applications. Among the multiple applications stand out the following cases:

- Inventory: It is probably the most important and promising RFID application. A special case of study is the Wal-Marts Race for RFID. Over the last decade, Wal-Mart has required its top 100 suppliers to use RFID tags on cases and pallettes of consumer goods. Wal-mart adopted the EPC standard, and in collaboration with the ID-center was one of the main promoters of the standard. At the beginning of the implantation, Wal-mart had to face some privacy issues related with RFID. More recently, NASA has deployed its *Project RFID* for use on the International Space Station (ISS), which includes using an RFID reader with both barcoding and RFID capabilities [145].

- Tracking: The International Air Transport Association (IATA) analysed in 2005 the adoption of the RFID technology for the sorting and handling of baggage. The conclusions were a Win-Win-Win situation for the three main stakeholders, the airlines, the airports and the passengers. Nowadays RFID systems in airports are fundamental, not only for the sorting of baggage but also for other applications like e-tickets.[72]

- Payments: Since late 1980s, automated toll collection has been widely implemented in the highways around the world. Electronic toll systems use active RFID devices because their read range. These automated systems allow the users to pay without stopping the car. The system helps to reduce traffic jams caused by tolls.

- Security: RFID door locking systems have been deployed in hotels and resorts which not only serve to grant secured access to a room but also to gain entrance to theme parks and other restricted areas of the resort. Nonetheless, new hotels are looking for innovative ways to differentiate themselves. Other new trending, where RFID is used, is to unlock a car door or start the car engine with a RFID card.

1.2 RFID standards

One of the key factors to deploy a new technology is standardization. RFID standards are guidelines or specifications for all RFID products. Standards provide guidelines about how RFID systems work, what frequencies they operate at, how data is transferred, and how communication works between the reader and the tag. Multiple standards for RFID technology have been developed during the years. This diversity complicates the expansion of the technology.

There are two main actors around RFID technology standardization, ISO and EPC Global. Many organizations making standards around RFID base their standards on existing ones developed by the ISO or EPC, and then they present a tailored solution for their application needs.

1.2.1 ISO standards

Internacional Organization for Standardization (ISO) is the largest developer of technical standards in the world. In 1996 it set up a joint committee with Internacional Electrotechnical Commission (IEC) to look at standardisation for RFID technology. ISO has developed a variety of standards related to RFID technology covering aspects such as the air interface, communication protocols, data control and formatting, applications and other smallest areas. Some standards related to RFID technology developed by ISO [56] are following presented:

- **ISO standards developed for identification cards:**
  - ISO 10536 Identification cards ; Contactless integrated circuit cards ; Close-coupled cards.

  - ISO 14443 Identification cards ; Contactless integrated circuit cards ; Proximity cards.
  - ISO 15693 Identification cards ; Contactless integrated circuit cards ; Vicinity cards

- **ISO standards developed for unit management level:**
  - ISO 15962 Radio frequency identification (RFID) for item management.
  - ISO 15963 Information technology ; Radio frequency identification for item management ; Unique identification for RF tags.
  - ISO 19762 Information technology ; Automatic identification and data capture (AIDC) techniques.

RFID standards have been developed by the following committees: ISO JTC1 SC31, ISO JTC1 SC17, ISO TC 104 / SC 4, ISO TC 23 / SC 19, ISO TC 204 and ISO TC 122. These committees developed the following standards widely used in dayly RFID systems:

- ISO 6346/9897/10374/18185/23389 Freight containers.
- ISO 9798/15434/15961/15962/15963 Information technology.
- ISO 11784/11785/14223 Radio frequency identification of animals.
- ISO 14816 Road transport and traffic telematics ; Automatic vehicle and equipment identification ; Numbering and data structure.
- ISO 17358/17363/17364/17365/17366/17367 Supply Chain RFID Standards.
- ISO 18000 Information technology ; Radio frequency identification for item management-
  - ISO 18000-1 Reference architecture and definition of parameters to be standardized
  - ISO 18000-2 Parameters for air interface communications at 135 KHz
  - ISO 18000-3 Parameters for air interface communications at 13,56 MHz
  - ISO 18000-4 Parameters for air interface communications at 2,45 GHz

- ISO 18000-5 Parameters for air interface communications at 5,8 GHz
- ISO 18000-6 Parameters for air interface communications at 860-930 MHz
- ISO 18000-7 Parameters for air interface communications at 433.92 MHz

1.2.2 EPC standard

EPCglobal is leading the development of industry-driven standards for the Electronic Product Code (EPC) to support the use of Radio Frequency Identification in today's fast-moving, information rich, trading networks.

Auto-ID Center was the responsible of EPC creation and development. Auto-ID Center developed an UHF protocol. Originally, this protocol was intended to be used with all kind of tags. The tags contemplated by the Auto-ID center were classified by their complexity:

- Class 0: UHFl read-only, preprogrammed passive tag.
- Class 1: UHF or HF; write once, read many.
- Class 2: Passive read-write tags that can be written to at any point in the supply chain.
- Class 3: Read-write with on-board sensors capable of recording parameters like temperature, pressure, and motion; can be semi-passive or active.
- Class 4: Read-write active tags with integrated transmitters; can communicate with other tags and readers.
- Class 5: Similar to Class 4 tags but with additional functionality; can provide power to other tags and communicate with devices other than readers.

RFID Class-1 Generation-2 systems are the most used. *Class-1*, as aforesaid, refers to the functionality of the tag while *Gen-2* refers to the physical and logical standards of tag and the encompassing system. *Gen-2* tags are used for item level identification in retail environments.

GS1's EPC *Gen2* air interface protocol, first published by EPCglobal in 2004, defines the physical and logical requirements for an RFID system of interrogators and

passive tags, operating in the 860 MHz - 960 MHz UHF range. Over the past decade, EPC Gen2 has established itself as the standard for UHF implementations across multiple sectors, and is at the heart of more and more RFID implementations [47]. Following the most relevant characteristics establish in EPC-C1G2 are presented:

1.2.2.1 Layers

EPC C1G2 establishes two layers:

- **Physical layer:** In this layer, the supported modulation techniques are specified: Double-sideband amplitude shift keying (DSB-ASK), single - sideband amplitude shift keying (SSB-ASK) , or phase-reversal amplitude shift keying (PR-ASK). Moreover it is included the coding allowed the encoding format, selected in response to Interrogator commands (FM0 or Miller - modulated subcarrier). Finally, it establishes that the communication link between reader an tag is half-duplex.

- **Tag-identification layer:** An Interrogator manages Tag populations using three basic operations [47]:

  a) Select. Choosing a Tag population. An Interrogator may use a *Select* command to select one or more Tags based on a value or values in Tag memory, and may use a *Challenge* command to challenge one or more Tags based on Tag support for the desired cryptographic suite and authentication type. An Interrogator may subsequently inventory and access the chosen Tag(s).

  b) Inventory. Identifying individual Tags. An Interrogator begins an inventory round by transmitting a *Query* command in one of four sessions. One or more Tags may reply. The Interrogator detects a single tag reply and requests the Tag's EPC. Inventory comprises multiple commands. An inventory round operates in one and only one session at a time.

  c) Access. Communicating with an identified Tag. The Interrogator may perform a core operation such as reading, writing, locking, or killing the Tag; a security related operation such as authenticating the Tag; or a file related operation such as opening a particular file in the Tag's User

memory. Access comprises multiple commands. An Interrogator may only access a uniquely identified Tag.

1.2.2.2 Memory Distribution

Tag memory shall be logically separated into the four distinct memory banks, each of which may comprise zero or more memory words. The memory banks are [47]:

- **Reserved memory** shall contain the kill and/or access passwords, if passwords are implemented on the Tag. The kill password shall be stored at memory addresses 00_h to $1F_h$; the access password shall be stored at memory a d dresses 20_h to $3F_h$.
- **EPC memory** shall contain a Stored-CRC at memory addresses 00_h to $0F_h$, a Stored-PC at addresses 10_h to $1F_h$, a code (such as an EPC, and hereafter referred to as an EPC) that identifies the object to which the Tag is or will be attached beginning at address 20_h, and if the Tag implements Extended Protocol Control (XPC) then either one or two XPC word(s) beginning at address 210_h.
- **TID memory** shall contain an 8-bit ISO/IEC 15963 allocation class identifier at memory locations 00_h to 07_h. TID memory shall contain sufficient identifying information above 07_h for an Interrogator to uniquely identify the custom commands and/or optional features that a Tag supports.
- **User memory** is optional. If a Tag implements User memory then it may partition the User memory into one or more files.

1.2.2.3 Security

As information flows between the reader and tags, EPC C1G2 standard establishes that tags shall implement a random or pseudo-random number generator (RN16). RN16 must meet the following requirements [47]:

- The probability that any RN16 drawn from the PRNG has value $RN16 = j$ for any j, shall be bounded by $0.8/2^{16} < P(RN16 = j) < 1.25/2^{16}$.

Table 1.2: EPC C1G2 tags main properties [121]

Identification	96 bit
Communication range	$\sim$ 5 m
Tag power consumption	$\sim$ 10 μ W [20]
Frequency (Europe)	865-868 MHz(UHF)
Operation frequency	100 KHz [21]
Tags Tx ratio	40-640 kbps
Tags Rx ratio	26.7-128 kbps
Identification per second	200
Area devoted to security	4000 Gates Equivalents [111]

- For a tag population of up to 10,000 tags, the probability that any two or more tags simultaneously generate the same sequence of $RN16$ shall be less than 0.1%, regardless of when the tags are energized.
- An $RN16$ drawn from a tag's PRNG 10ms after the end of T_r (RF signal envelope rise time) shall not be predictable with a probability greater than 0.025% if the outcomes of prior draws from PRNG, performed under identical conditions, are known.

In addition, a tag may implement a security timeout after a failed command. Although this protocol gives Tag manufacturers the option of choosing which commands are subject to a security timeout, it recommends that Tags implement a security timeout at least for the Access-command sequence.

EPC Gen2 tags are passive and power dependent from the reader to respond the queries. Table 1.2 summarizes the main characteristics of these tags.

1.3 RFID Security

RFID problems related to security are similar to those present in computers and networks. However, RFID technology has two important handicaps. The first handicap is related to the resources devoted to security in the tag. These resources, mainly area, are directly connected to the tag cost. The second handicap in order to provide security to RFID systems is the wireless nature of the technology. It is easier for an attacker to eavesdrop information in a wireless system without being detected [7].

1.3.1 RFID security properties

Basic security requirements in RFID technology can be listed as follows:

- **Confidentiality:** Is the feature that guarantees that only accredited users can access the information. Confidentiality can be provided using physical protection or at algorithm level. Confidentiality is a key feature because RFID allows the tracking of items that suppose a threat to the users.
- **Integrity:** guarantees that the messages transmitted between two parties are not modified in the communication process. Integrity includes insertion, deletion, and substitution. This feature is specially important in re-writeable tags. Typically, CRCs that only protect against random changes are used.
- **Availability:**It is the capacity of being available for accredited users. Some RFID systems are easily disturbed using frequency interferences. Denial of Service (DOS) attacks are the biggest threat against availability.
- **Authenticity:** It is important to guarantee that the tag can not be cloned or falsified.
- **Privacy:** As RFID can be tracked, it is important to assure that the tag owner can not be tracked in time or space.

1.3.2 Attacking RFID systems

One of the main concerns about RFID technology is security, due to the fact that with a simple reader tuned to the appropriate frequency, information can be eavesdropped. It is important to know the different threats against these devices in order to prevent the attacks. Some of the usual attacks are the following [48]:

- **Spoofing:** Spoofing attacks supply false information that looks valid and the system accepts. In RFID systems, typically broadcasts an incorrect EPC ID when a valid ID is expected.
- **Insert attacks:** Insert attacks consist in the insertion of system commands where data are generally expected.

- **DOS attacks:** Denied of Service (DOS) attacks are carried out when a signal transmits more information than it can be handled.
- **Deactivation:** Deactivation is an attack to the availability of the tag. It can be performed if the *deactivation code* is known.
- **Cloning:** This kind of attack is classified as an integrity attack. The attacker eavesdrops the tag identification and replicates this data in a fake tag.
- **Tracking attacks:** This attack affects the privacy of the user tag. The tracking can be in time or in space.
- **Replay attacks:** It is an integrity attack where some traces that had been observed in previous communications are sent.
- **Man in the Middle (MitM):** MitM is an integrity attack where an attacker tries to impersonate a legitimate reader, adding, modifying or deleting original traces of the communication.

1.3.3 RFID algorithms

In previous sections, the importance of authentication in RFID systems has been depicted. Authentication protocols can be classified according to the complexity of the operations involved in the computation [79][34]. Four groups can be listed:

- **Full-fledged:** These kind of protocols allow the implementation of classic cryptography as public key or symmetric encryption. These protocols are used in secure demanding applications like E-passports. Some full-fledged protocols are presented in [5, 115, 126].
- **Simple:** These protocols accept random number generation and hash function but public key encryption is not allowed. Two RFID protocols classified in this group are [4, 104]
- **Lightweight:** These protocols support the generation of random number, simple functions as Cyclic Redundancy Check (CRC) or bitwise operations.
- **Ultra-lightweight:** Only bitwise operations are allowed (XOR,AND,OR).

1.3.4 Low-cost RFID security

Typical cryptographic blocks developed for modern cryptography are not suitable for low-cost tags. These blocks are based on difficult mathematical challenges that involve intensive computational operations. In the last decade, an emerging field known as Lightweight cryptography has been developed. Lightweight cryptography involves several disciplines like cryptography, information technology, RF engineering and microelectronics.

Lightweight cryptography comprehends the algorithms designed to be suitable for RFID low-cost tags. As aforementioned, these kind of tags have very limited resources (storage,area,power consumption,etc). These resource-constrained environments make providing security in RFID a challenging task.

It is necessary to reduce the tags cost in order to deploy them massively. Used silicon area is usually directly related to the costs. In RFID systems, the area devoted to security in tags is up to 4000 GE. When this constraint is compared with the approximate 8120 GE [51, 102] required by a standard hash function like SHA-1 (which is an essential building block for most security protocols), it becomes clear the need for schemes that can provide some minimum security services while requiring as few resources as possible.

Power consumption is also a constraint imposed by the system. As RFID tags are passive, it is commonly accepted that only 10 μW can be consumed [20]. This limit conditions the number of operations that can be carried out in parallel and also the complexity of these operations (number of blocks involved in the computation).

Tags per second rate is also a constraint to handle in protocols design. Typical applications demand up to 200 Tag/sec, that supposes between 500-600 clock cycles (at 100 KHz) to compute the entire protocol [87, 94].

Finally, before designing a new protocol it is important to take into consideration the final application. For example, an RFID E-passport does not require the same security level as a low-cost tag employed in the supply chain (i.e. tags conforming to the EPC-C1G2 specification). In the specific case of this thesis, focused on EPC-C1G2 tags, the standard establishes its own security requirements like the integration of an RN16 or the Tag-identification layer.

Since in 2003, Vajda and Buttyan introduced a set of challenge and response

lightweight authentication protocols ([136]), several proposals have been presented in the literature. Among these proposals, algorithms specifically oriented to EPC-C1G2 tags ([105][94],etc) are included. The main weakness of these proposals is the lack of a realistic implementation and hardware results. In this thesis, the design and implementation of lightweight cryptographic algorithms suitable for EPC-C1G2 tags has been addressed from a realistic point of view.

2

Hardware Footprint Estimation of Lightweight Cryptographic Primitives

A major difficulty in providing RFID tags with security functions comes from the scarcity of computational resources available in such platforms (see, e.g. [140, 125, 13] for recent developments in the design of tiny tags). For example, it is commonly assumed that only between 250 and 4K Gate Equivalents (GE) can be devoted to security functions in a low-cost RFID tag [70], which restricts affordable solutions to lightweight algorithms only.

Typically, cryptographic algorithms have been designed to be efficient in software. This is due to the fact that they were intended to be used in commercial applications like PCs.

Nowadays, cryptographic modules can be implemented efficiently either by hardware or by software. On the one hand, software implementations are known for being easier to be developed and maintained. In addition, with the development of custom microprocessor instructions, these software implementations use microprocessor resources efficiently.

On the other hand, for cryptographic modules or security-related applications in general, software implementations are significantly less secure than their hardware equivalents. The reason for this is mostly the fact that software solutions make use of shared memory space and are running on operating systems. Moreover there are some resource-constrained applications, like low-cost RFID tags, that can not use software cryptographic modules because it is not possible to integrate a microprocessor.

Hardware proposals in the area of lightweight cryptography have proliferated over the last years, but most of them do not provide information about their implementation:

- Very theoretical contributions that do not provide any proof of their lightweightness are reported in the literature [122] [110].
- In many cases, arguments in favor of their lightweightness are based on the use of some operations that are generally considered inexpensive by the authors. However, these estimations are not always correct, and the implementation of some proposals greatly exceeds the area limit of 4K GE [73][97][34].
- In other cases, the design turns out to be not so lightweight because of factors such as the bit length of the variables, the need for additional memory blocks –which is usually missed in the analysis of resources–, and the overhead impossed by selection and control logic. These and other aspects often make the final gate count much higher than expected [34][110].

All in all, providing accurate estimations of the footprint area of an ASIC implementation is a hard task for algorithm designers [66, 59]. More often than not, designers do not have the hardware design skills nor the tools required to implement and analyze their proposals. Furthermore, there is a lack of a standard methodology to provide such an estimation, as the result will vary depending on factors such as the chosen architecture, the manufacturing technology, the possibility of introducing optimizations at various levels, etc. Problems such as these are common in other related areas. For example, designing low-power embedded systems [81, 152] is also a major problem nowadays, and system designers face a situation similar to that described above. Very recently, Ben Atitallah *et al.* [15] have presented a methodology to provide designers with an estimation of the power consumption of a complete system. Similarly, in this chapter we propose a relatively simple estimator for the footprint area occupied by the ASIC implementation of an algorithm. The suggested formula requires the designer to know only a few high-level details about the target implementation, such as the number of registers used to store inputs, outputs and intermediate variables, and some parameters related to the control structures. Our work is motivated by and focussed on lightweight cryptographic algorithms for constrained platforms such as RFID tags or sensor nodes.

The rest of this chapter is organized as follows. In Section 2.1 we provide an overview of the main basic elements that are used in lightweight algorithms. In Section 2.1.2 we show hardware architectures usually employed to implement these elements. In Section 2.2, the Area estimator is introduced. First of all, the area results of basic operation blocks using two real manufacturing libraries are depicted in subsection 2.2.1. Subsequently, in Section 2.2.2, we present a method to estimate the footprint area of a whole algorithm and in section 2.2.3 we discuss our experimental results with a battery of real-world examples. Finally, Section 2.3 concludes the chapter and summarizes our main contributions.

2.1 Study of Lightweight Cryptographic primitives

2.1.1 Elements in Lightweight Cryptography

In this section, we provide a brief description of the usual operations found in lightweight cryptographic primitives and protocols. This will serve to motivate our subsequent footprint analysis for individual building blocks.

2.1.1.1 Cryptographic Operations

As in the case of regular security functions, lightweight cryptographic primitives and protocols aim at providing basic constructions to guarantee properties such as the confidentiality, integrity and authenticity of data exchanged in communications. In this case, however, the shortage of resources in the platforms severely limits the sort of processing that can be done. Thus, most proposals attempt to rely only on a few simple bitwise operations (such as, for example, XOR, OR, AND, shifts and rotations) and inexpensive arithmetics such as addition modulo 2^m.

- **Triangular functions: XOR, AND, OR, Addition and Multiplication**
  In 2004, Klimov and Shamir introduced the concept of triangular functions (T-functions) [76], which encompass most of such operations. A T-function is a mapping from m-bit words to m-bit words where for each $0 \leq i < m$, the

i-th bit of the output only depends on input bits $0, 1, \ldots, i$. Bitwise operations, such as for example XOR ($a \oplus b$), OR ($a \vee b$), AND ($a \wedge b$) and many others found in modern processors (e.g., addition $a + b \bmod 2^m$ and multiplication $a \cdot b \bmod 2^m$) are T-functions, and their compositions are T-functions too [76].

Secure cryptographic primitives and protocols cannot be designed by exclusively using T-functions. A T-function has poor diffusion as it does not propagate information from right to left, and its period is predictable [10]. As a consequence, T-functions are not the only operations that are usually found on lightweight cryptographic algorithms. For instance, in [34, 39] some arguments are given for mixing triangular and non-triangular functions in order to design more secure ultra-lightweight protocols.

- **Rotation**

  One of the most common non-triangular function used in cryptography is the rotation operation. Rotation can be performed in several ways. For instance, $rot^*(x, y)$ represents a circular shift of x by $wht(y)$ positions to the left, where $wht(y)$ is the Hamming weight of y. In the classical definition, $rot(x, y)$ is a circular shift of x by $y \bmod N$ positions to the left, where N represents the bit length of variables x and y. Choosing a particular N determines the lightweight nature of this operation. For example, if N is a multiple of 2^n, then $rot()$ can be implemented very efficiently since it reduces to shifting the first argument n positions to the right; otherwise, it becomes more complex and requires a larger footprint area.

2.1.1.2 Storage and Control: Registers and multiplexers

Apart from arithmetic and logical operations, algorithms also require additional hardware resources to store results and to control the execution flow. Such elements are nearly always registers and multiplexers. Registers help to maintain the "state" of the algorithm by storing intermediate and final results, and also by supporting the control functions. Multiplexers are used to select among different inputs according to some conditions, and are instrumental in any algorithm that incorporates a minimum flow complexity (i.e., loops, ifs, etc.).

In general, the area occupied by registers and multiplexers is critical, and any

good design should find a balance between the amount of basic operations and memory/control logic. It is quite common to find proposals where authors only analyze the complexity of their designs by counting the amount of operations. While this might be useful to determine the computational complexity of the algorithm, ignoring memory and control requirements could be very misleading in terms of the footprint area of the circuit. In fact, in many cases the area required by these elements is much larger than that demanded by the arithmetic and logical components.

2.1.2 Hardware Implementation of Basic Operations

It is well-known that there are several ways of implementing a circuit. Serial, parallel and Pipeline approximations are the most recognized.

Serial implementations use one common operation block sharing it to carry out each operation. Serial implementations are generally the most efficient in area and have a big penalty in throughput.

In parallel implementations, several operations are carried out simultaneously. These kind of implementations often use less clock cycles but have a penalty in area due to the necessity of replicating blocks.

Finally, the increasing demand for high speed ASICs is driving the requirement to increase circuit throughput in terms of calculations per clock cycle. The performance of an ASIC can be increased by pipelining but at an expense of increase in system latency and area.

We next analyze various design elements and their area estimation depending on the complexity of their implementation. For low-complexity blocks, we propose a simple architecture, while for those with higher complexity we study and propose several possible architectures.

2.1.2.1 Elements with Low Complexity Implementation

In this group we include some T-functions such as simple bitwise operations and addition $mod\ 2^m$. Registers and multiplexers are also in this category. Each of these elements has a straightforward implementation with very low complexity.

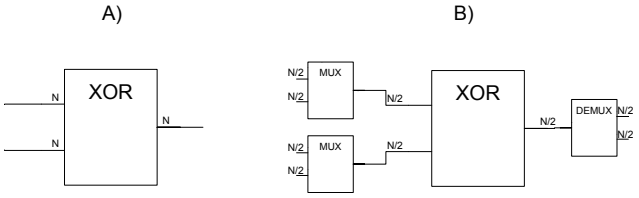


Figure 2.1: Architectures for an XOR block.

It is common to find implementations for 96 bits, as it is a common bit length widely used in low-cost RFID tags such as those conforming to EPC Class-1 Gen-2 standard (ratified as ISO/IEC 18000-6C). We study the low complexity elements for different number of bits, including among them 96-bits, obtaining a constant K for each basic element. These constants will be subsequently used to obtain area estimations for more complex constructions.

For the above mentioned bitwise operations, we can use a strategy to reduce the area cost with a penalty in throughput. This strategy consists in reducing the bit length of the block and using several clock cycles to obtain the final result. For example, in an algorithm that uses an N-bit XOR block, it can be reduced to a block of $N/2$ bits but needing two clock cycles to obtain the result. We note that the reduction of a block involves the use of additional multiplexers for the control logic. Thus, we need to find a trade-off between the reduction and the necessary extra logic. Moreover, the drop in throughput has to be taken into consideration, as it should always meet the restrictions imposed by the operational environment. For instance, taking as reference the performance criteria of an RFID system that demands a minimum reading speed of at least 150 tags per second [35, 21], we need to carefully calculate the reduction of the block to fulfil this reading rate requirement. Note, that this reduction strategy can be used with other operations such as adders and the rest of bitwise operations.

For illustration purpuses, in Fig. 2.1(a) we show the scheme of an N-bit XOR block. Fig. 2.1(b) shows the result after halving the XOR block ($N/2$ bits) and introducing additional multiplexers to select inputs and outputs.

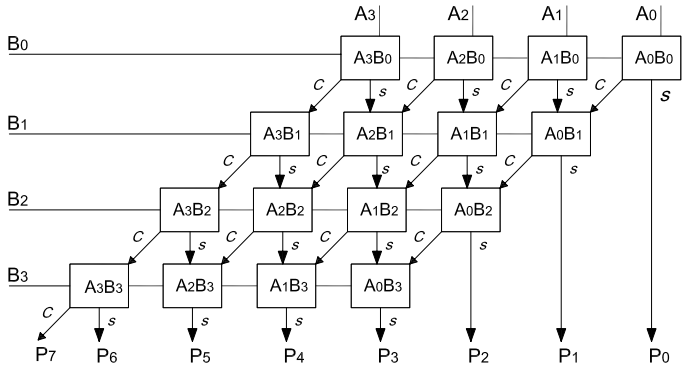


Figure 2.2: Combinational Multiplier Architecture

2.1.2.2 Multiplication Operation

In this section we study the multiplication operation. There are several possible hardware architectures for its implementation, including: 1) a combinational architecture; 2) a shift-and-add architecture; and 3) the Karatsuba-Ofman architecture [98, 123].

1) **Classical Combinational Multiplication**

The best implementation, in term of clock cycles, of a multiplier is achieved by exclusively using combinational logic. Considering the partial multiplications that are needed, the multiplier can be implemented with appropriate logic for each partial multiplication, and adders to perform the addition of the partial multiplications. It should be noted that this approach has the disadvantage of having a high cost in area. As an example, the architecture required for 4-bit unsigned binary numbers is shown in Fig. 2.2.

In Fig. 2.3, we show the internal structure of a basic cell. This cell includes a Full Adder (FA) and an AND gate. The AND gate computes the product of each bit of the multiplier q_j with the corresponding bit of the multiplicand m_j. The output of this product is one of the inputs to the FA. The remaining operands are the corresponding bit from the previous partial product (Pp_i) and the carry (c) generated in the previous stage.

2) **Classical Shift and Add Multiplication**

Partial products can be stored in a register and multiplications can be obtained

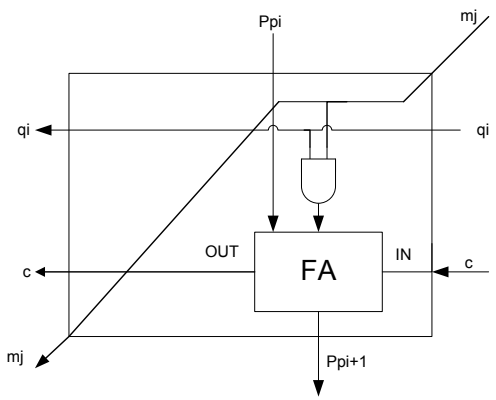


Figure 2.3: Basic Cell of Combinational Multiplier

in several clock cycles in order to reduce the necessary hardware. To optimize the number of clock cycles, the considered partial multiplications are the multiplicand times each bit of the multiplier (i.e., there are as many partial multiplications as bits have the multiplier). As multiplication times two can be implemented as a shift left, the partial multiplications can be implemented with just a shifting left register. An adder is necessary for the addition of the partial results, and some control logic completes the architecture of this multiplier as illustrated in Fig. 2.4. This sequential multiplier uses very few hardware resources, but it is slow because many clock cycles are required. If we consider big operands, this architecture uses N cycles, where N is the bit length of the operands.

3) **Karatsuba-Ofman**

Karatsuba Ofman [98] is a much more efficient algorithm in terms of its area/time factor. It is based on a divide and conquer strategy. A $2n$-digit integer multiplication is reduced to two n-digits multiplications, one $(n+1)$ digits multiplication, two $n-$digits subtractions and two $2n$-digit additions.

We consider the product $X \times Y$ of 2 integers, X and Y. These integers can be split into two halves (i.e., X_H, X_L, and Y_H, Y_L). Let n be the number of bits

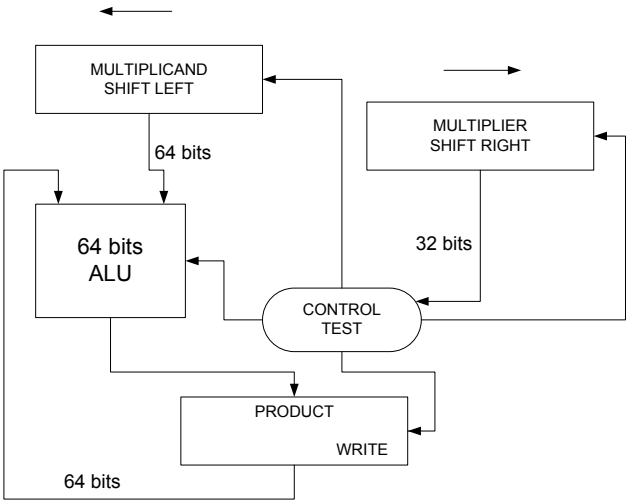


Figure 2.4: Shift and Add Multiplier

of each of these halves:

$$X = X_H \cdot 2^n + X_L \tag{2.1}$$
$$Y = Y_H \cdot 2^n + Y_L \tag{2.2}$$

The product $P = X \times Y$ can be obtained by computing four n-bit multiplications:

$$\begin{aligned} P &= X \times Y = (X_H \cdot 2^n + X_L)(Y_H \cdot 2^n + Y_L) = \\ &= 2^{2n}(X_H Y_H) + 2^n(X_H Y_L + X_L Y_H) + X_L Y_L \end{aligned} \tag{2.3}$$

Finally, the computation of (2.3) can be improved by applying the equality:

$$\begin{aligned} X_H Y_L + X_L Y_H &= \\ (X_H + X_L)(Y_H + Y_L) - X_H Y_H - X_L Y_L \end{aligned} \tag{2.4}$$

Summarizing, the $2n$-bit multiplication $(X \times Y)$ can be thus reduced to three n-bit multiplications: $X_H Y_H$, $X_L Y_L$ and $(X_H + X_L)(Y_H + Y_L)$ by applying the Karatsuba-Ofman algorithm. In fact, the algorithm performs a multiplication operation by using smaller multiplications and some adders. Different design

architectures can be considered for the implementation, by just selecting different implementations for these adders and multipliers, which have already been mentioned in the previous sections.

Specifically, we study two architectures. The first one uses a combinational multiplication of n bits. This architecture computes each multiplication in one clock cycle. The second architecture uses a shift and add multiplication of n bits, with the penalty of using several clock cycles in each multiplication.

2.1.2.3 Modulo Reduction

This operation is commonly used in modular multiplications or even in rotations like $rot^*(x, y)$. The hardware needed for its implementation depends on the value of the modulo. Given two positive numbers P (dividend) and N (divisor), $P \ mod \ N$ outputs the remainder of the division of P by N. This operation is considered lightweight when N is a multiple of 2^n because the division can be executed by shifting to the right. Nevertheless, N may not be a multiple of 2^n and for this reason we present an implementation that allows the computation of the modulo for each value N.

The modulo operation requires the computation of a division, which is a more complex operation than the multiplication. One straightforward algorithm is the Naive Reduction. This algorithm shifts and subtracts the modulus until the remainder is obtained. A subtractor, a comparator and n-bit register are the only the hardware needed for its implementation, but it uses a large amount of clock cycles (2^n). Alternatively, if we are wealthy in resources we can use the Non Restoring Reduction (see Fig. 2.5), which is much more efficient but uses more hardware. In this case, the modulo is obtained in approximately n steps.

Sometimes, a combination of a multiplication and a modulo reduction can be found in some algorithms. There are special implementations to optimize these combinations, usually involving the Montgomery modular multiplication scheme and different possible architectures [68]. As this kind of operations are often not lightweight, we will not study them in detail in this chapter.

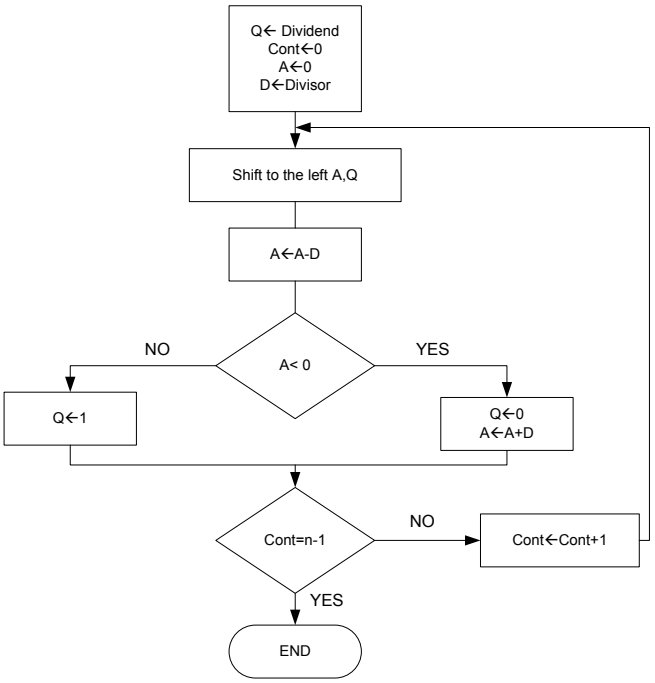


Figure 2.5: Non Restoring Reduction Algorithm

2.2 Estimating the Area of Lightweight Algorithms

2.2.1 Previous considerations

As discussed above, one major goal of this work is to provide an estimation of the area required by a lightweight cryptographic algorithm as a function of some high-level parameters. In these applications, it is crucial to keep in mind that circuits predominantly operate at low frequencies. For instance, many RFID tags function at 100 KHz. (Note that the 100 kHz frequency refers to the clock included in the tag circuit, not to the communication band that is generally in the 860-960 MHz range for EPC C1-G2 tags.) As the clock frequency is fixed, most restrictions in these designs relate to area and power consumption.

In this subsection, we report area results obtained with two specific manufacturing libraries for the different elements presented above. With these results, it will be possible to distinguish which elements can be regarded as a lightweight element. In addition, the area per bit (K_i) for each element, that will be used in the estimator,

is calculated. A priori, it is unclear to us the extent to which our conclusions generalize to other manufacturing technologies. The results, however, are still useful to compare different algorithms and classify them as lightweight or not. Furthermore, the methodology is general and can be easily extended to other libraries.

2.2.1.1 Synthesizer Setup

The experimentation has been conducted with two CMOS libraries: Faraday UMC 90 nm LL, tt 1.25V [134] and AMIS UCASCB 0.35 μm, tt 1.32V [1], where "tt" stands for *typical-typical* cells. One key reason behind this decision is that these libraries provide comprehensive information about the layout of basic cells. For our purposes, this is essential to obtain a realistic estimation of the area occupied by an algorithm. To synthesize each design we used Synopsys [2], which is one of the most commonly used tools.

The operation frequency is set to 100 KHz. As mentioned before, this is quite a common value for passive RFID tags. As for the synthesis with Synopsys, after experimenting with different configurations we observed that the best results are obtained with the *medium effort* option in area, delay and power consumption. These options are set for all the experiments.

Finally, the area results are provided using Gate Equivalents (GEs), which is the normalization commonly used for these applications. Using GEs facilitates comparisons among different implementations since the obtained values are independent of the chosen technology. The GE value is obtained by dividing the whole area of the circuit by the area of a basic NAND gate. For example, 1 GE for the UMC 90 nm takes 3.16 μm^2.

2.2.1.2 Area for Low-complexity Elements

Table 2.1 summarizes the area results (in GEs) obtained after synthesizing with Synopsis the set of basic elements for UMC 90 nm and AMIS 0.35 μm libraries. In this first analysis, the hardware architecture considered performs all operations (combinational or just registers) in one clock cycle. As shown in Fig. 2.6, the area occupied by each element increases linearly with the length (in bits) of variables. The results obtained for the AMIS 0.35 μm library are almost equivalent and follow a

Table 2.1: GEs for low-complexity elements

Lib.	Element	32 bits	64 bits	96 bits	128 bits	K_i
UMC 90nm	AND	39.70	79.39	119.08	158.78	K_1=1.24
	OR	39.70	79.39	119.08	158.78	K_2=1.24
	XOR	79.39	158.78	238.17	317.56	K_3=2.48
	ADD	239.66	477.84	716.01	954.19	K_4=7.45
	Multiplexer	71.00	143.00	214.00	285.81	K_{mux}=2.23
	Register	147.00	287.00	441.00	588.24	K_{reg}=4.59
AMIS 0.35 μm	AND	42.66	85.33	127.99	170.64	K_1=1.33
	OR	53.33	106.65	160.03	213.22	K_2=1.66
	XOR	74.66	149.34	224.00	298.70	K_3=2.33
	ADD	203.35	406.00	608.69	811.32	K_4=6.34
	Multiplexer	85.33	170.66	256.00	341.55	K_{mux}=2.66
	Register	214.33	435.66	651.66	869.00	K_{reg}=6.77

similar pattern. This simplifies considerably the analysis of more complex algorithms, as it allows us to associate a constant value, named K_i for element i, giving the area per bit for each element.

Some conclusions can be drawn from these results:

1. The adder occupies significantly more area than bitwise operations. Consequently, if the area of an algorithm needs to be optimized, it is more appropriate to focus on additions rather than concentrating on low complexity elements such as bitwise operations. As all operations are done in one clock cycle, one possibility to optimize the area would be to use an element with lower bit length and carry out the operation in various clock cycles. For example, variables can be split into two parts with half of the bits each and a half-length adder can then be applied over each part. Note, however, that in doing this we need to include additional elements, namely registers to store partial results, multiplexers to choose among different signals, etc.

2. The area cost of registers is also noticeable. Taking into account that we generally can devote just a small area to security subsystems (e.g., up to 4K GEs in most passive RFID tags), and that roughly 50% of it is used for storage, this means that at most five 96-bit registers could be used.

3. As for multiplexers, their cost in terms of area is small. These elements are needed in algorithms with loops (e.g., "for" and "while" iterations) and also when a input/output is selected among different signals.

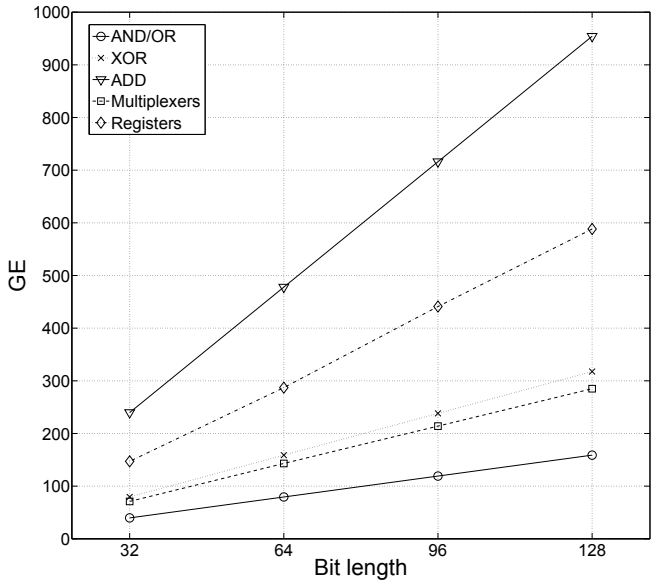


Figure 2.6: GE for low complexity elements as a function of the number of bits for UMC 90nm library.

Overall, it can be concluded that designers will necessarily face some trade-offs among operations and the amount of registers and multiplexers required. As a general rule, bigger building blocks (i.e., using a larger bit length) will require less extra registers/multiplexers, and vice versa.

2.2.1.3 Further Operations: Multiplication and Modulo Reduction

We next explore the area required by two operations that have been extensively used in many cryptographic algorithms: multiplication and reduction modulo N. The figures, both the number of GEs and the associated clock cycles required to complete the operation, are shown in Table 2.2 for the UMC 90 nm and the AMIS 0.35 μm libraries.

In general terms, multiplication cannot be considered as a lightweight operation no matter what architecture is chosen, since it demands more than the 4K GEs often required in environments such as RFID systems (96 bits). That being said, it is worth-noting that some trade-offs also appear here. The combinational architecture offers the best performance speed-wise, but it demands too much area. Conversely, the Shift-and-Add option is much more efficient in terms of area, but the number of clock cycles requires may be prohibitive for many applications. K-O architectures

Table 2.2: GEs for different multiplication architectures and modulo reduction.

Lib.	Operation	32 bits	64 bits	96 bits	128 bits	Cycles
UMC 90nm	MULT (Comb.)	9345	36507	81268	144452	1
	MULT (S+A*)	2078	4113	6146	8164	N
	MULT (K-O†, Comb)	5744	16055	30853	49868	10
	MULT (K-O†, S+A*)	4731	9367	13995	18566	$\frac{N}{2}+4$
	Modulo reduction	–	–	3967	–	96
AMIS 0.35 μm	MULT (Comb.)	10223	36495	81007	143436	1
	MULT (S+A*)	2464	4840	7227	9639	N
	MULT (K-O†, Comb)	6621	17093	32123	51132	10
	MULT (K-O†, S+A*)	5882	11610	17341	23111	$\frac{N}{2}+4$
	Modulo reduction	–	–	4729	–	96

†K-O: Karatsuba-Ofman multiplier *S+A: Shift-Add multiplier

fall somewhere in between of these two alternatives.

Modulo reduction is a special case. As discussed before, it is very lightweight when the bit length N is a power of two, as it can be implemented simply as various right shifts. Otherwise, such as for example for $N = 96$, its area takes around 4K GEs. Thus, our recommendation is to include it only when the resources required by this operation can be reused in other parts of the algorithm.

2.2.2 A Linear Estimator

Estimating the area that an implementation of an algorithm can occupy is quite challenging because it depends on many factors: the architecture(s) chosen by the designer, the specific constraints, the manufacturing library, the basic cells used by the synthesis tool, etc. In this section, we first propose an expression that estimates the total area required by a hardware implementation of an algorithm. Subsequently, we check its validity by comparing its predictions with the actual area obtained with a battery of examples and provide a refinement of our estimator. Note that we have discarded the use of multiplication since this operation uses resources in excess (>4K GEs) to be categorized as a lightweight operation. Regarding modulo reduction,

its usage in a lightweight algorithm is conditioned to be a power of two (i.e. 2^n); otherwise it demands more than 4K GEs and using it is unfeasible. In case of being a power of two, the operation does not use any extra hardware resources, but requires an upper bound of n clock cycles to compute it.

The total area occupied by an algorithm can be roughly divided into two main blocks: datapath and control. The datapath contains the hardware for the different operations required and registers to store inputs, outputs and intermediate results. In many lightweight cryptographic algorithms, the datapath accounts for a significant fraction of the total area, generally around 80% [105] [94].

Our estimation is based on the following rationale. As we previously pointed out, the final footprint depends on the chosen architecture. In turn, opting for one architecture or another depends on the goals and restrictions faced by the designer. For example, on very constrained devices (such as RFID tags or some sensor nodes) minimizing the area is a priority, which heavily influences the decision. Since throughput is often a limiting factor too, one sensible choice is an architecture that optimizes the area without penalizing throughput too much. In general, such a design contains one single block of N bits for each basic operation needed, plus registers to store data and multiplexers to select inputs and outputs.

Based on the previous considerations, we propose a simple linear estimation for the area of the datapath, measured in GE, as a function of the bit length and the number of basic operations, registers and multiplexers:

$$F_{DP} = N \cdot \left[\sum_{i=1}^{4} A_i \cdot K_i + (B \cdot K_{reg}) + \left((C + D) \cdot K_{mux} \right) \right] \tag{2.5}$$

where:

- N is the bit length of the variables.
- A_i is a parameter dependent on the chosen architecture for the datapath ($i = 1$ for AND, $i = 2$ for OR, $i = 3$ for XOR, and $i = 4$ for ADD). As discussed above, the implementation can range from a fully combinational design to one using smaller operators but requiring more clock cycles. Thus, we measure A_i as the number of N-bit operators.
- K_i is the area cost for the i-th operation, as shown in Table 2.1.

- B is the number of variables that require storage.
- K_{reg} is the area cost for each register.
- C is the number of multiplexers necessary to select different inputs for the operation blocks. When the block has more than two inputs, C is the number of inputs minus one.
- D is the number of multiplexers necessary to select different inputs for each register. If the algorithm is given in pseudocode, D can be easily estimated as the number of assignments made for each variable.
- K_{mux} is the area cost for the multiplexers.

Obviously, expression (2.5) only factors in those elements studied in Section 2.2.1. However, it can be extended without difficulty to any other blocks that conform to the design rationale given in the paragraph above.

Finally, as the area of datapath and control are in most cases related, we express the total area as:

$$F = (1 + \omega) \cdot F_{DP} \tag{2.6}$$

where ω is an *overhead* factor accounting for the control part (e.g., $\omega = 0.2$ assuming that control logic accounts for 20% of the total area).

2.2.3 Experimental results

We have tested our estimator against a library containing 120 lightweight functions. The algorithms are named $F_1, F_2, \ldots, F_{30}$ and were synthesized for four different bit lengths: $N = 32$, 64, 96, and 128 bits. Each function returns a single final output value denoted Z and uses several input and intermediate variables, represented by X_i and Y_i, respectively. The dataset is well balanced, containing 10 functions with 2 inputs, 10 functions with 4 inputs, and another 10 functions with 6 inputs. The data set of functions is depicted in A.

In Fig. 2.7 we compare the estimated area for all the datasets functions using (2.6), assuming a control overhead $\omega = 0.2$, versus the actual area given by Synopsys after synthesizing each function. For simplicity, we only show the results obtained

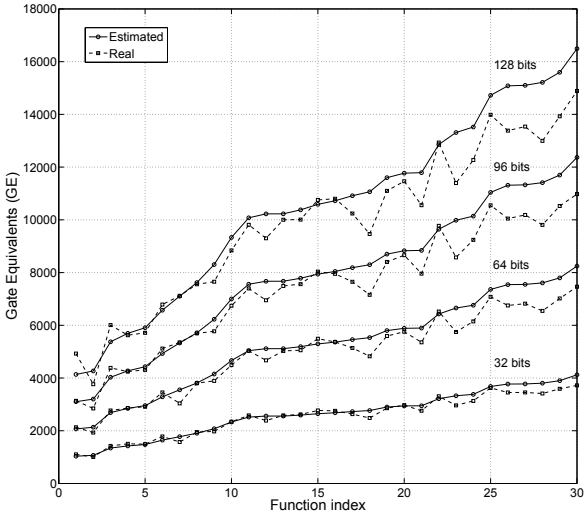


Figure 2.7: Real and estimated footprint area ($\omega = 0.2$).

for the UMC 90 nm library. As suggested by Tables 2.1 and 2.2, the results for the AMIS 0.35 μm are completely equivalent, and our experimentation confirms this. The approximation is quite precise, with differences becoming greater when the number of bits N increases. In Fig. 2.8 we show histograms of the estimation errors for different bit lengths. For $N = 32$ and 64 bits, errors are bounded by 500 GE and 1K GEs respectively. This error increases to 1.5K GE and 2K GEs for $N = 96$ and 128 bits, respectively. Thus, chosing a high value for the control overhead (20%) in Equation 2.6 does not minimize errors but guarantees an overestimation of circuit area.

Further investigations reveal that the overestimation does not come from the expression (2.5), but from (2.6). In other words, the estimation for the datapath area is fairly accurate, but the amount of control logic does not generally increase linearly with the number of bits. For instance, a Finite State Machine (FSM) controlling some parts of an algorithm does not need more states when variables increase their size. That being said, we emphasize that our choosing of (2.6) may still be valid for constrained designs, where N often varies between 32 and 512, considering the result as an upper bound.

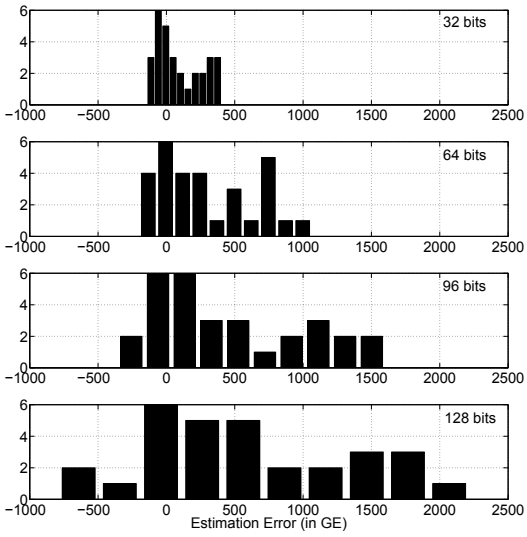


Figure 2.8: Distribution of gate count estimation errors ($\omega = 0.2$)

2.2.4 Adjusting control overheads

As discussed in Section 2.2.2, the datapath and control areas are in most cases related. In the model presented above we made the assumption that the relation is linear, in particular with the control logic being a fraction $(1 + \omega)$ of the datapath area. The experimental results discussed above show that this assumption works relatively well for systems of up to 10K GE, particularly with $\omega = 0.2$. The estimation error becomes more significant for bigger systems. This is reasonable, as an increase in the datapath footprint does not necessarily translate into a similar increase of control logic.

Using the dataset of designs described above, we have numerically investigated more precise approximations for the control overhead term used in (2.6). Two alternatives were explored, both based on the idea that ω varies with some system parameters. In the first one, we assumed that the control overhead depends on the number of bits N, so the total area is actually of the form:

$$F = \left[1 + \omega(N)\right] \cdot F_{DP} \tag{2.7}$$

whereas in the second alternative it is assumed that the amount of control logic is a

function of the datapath area:

$$F = \left[1 + \omega(F_{DP})\right] \cdot F_{DP} \tag{2.8}$$

The estimation of both functions $\omega(N)$ and $\omega(F_{DP})$ was done by couching the problem as a nonnegative least-squares curve fitting one of the form:

$$\min_{\Omega} \|C\Omega - d\|_2^2 \tag{2.9}$$

where $\Omega = (\omega_1, \ldots, \omega_k)^T$, with $\omega_i \geq 0$, represents the sought function discretized in k values. Matrix C and vector d contain, respectively, the *actual* datapath area and total area obtained after synthesis.

We split the dataset of designs into two subsets. The first one, used to estimate the overhead function (training) contains 80 randomly chosen (10 of each bit length) designs out the 120 available. The remaining 40 designs will be subsequently used to test the obtained estimator. Thus, each one of the 80 synthesized functions used for training gives an equation for (2.9). These 80 equations are grouped into k bins. In the case of $\omega(N)$, we chose $k = 4$ values (32, 64, 96, and 128 bits), whereas for $\omega(F_{DP})$ we grouped equations into $k = 7$ intervals with a 2K GE difference between each of them.

Using a standard numerical solver, we obtained the Ω-values shown in Table 2.3. Again, these figures correspond to the UMC 90 nm library; those obtained for the AMIS 0.35 μm are very similar. Such overheads represent the best fit, in a least-squares sense, for our experimental dataset. As observed, in both cases the actual overhead is always below the fixed $\omega = 0.2$ value that was used before. Furthermore, it decreases as circuits grow bigger, both in terms of N and in datapath area, which conforms to our previous intuition. For example, in systems with less than 4K GE the overhead accounts for 16%-19% of the datapath area, but it falls down to less than 10% when the datapath is 10K GE or more. This is also observed when the overhead is considered a function of N.

Analysis of the squared 2-norm of the residual reveals that the $\omega(F_{DP})$ estimation performs significantly better than $\omega(N)$. Thus, while the former yields a squared residual of 1.38E+05, which roughly translates into an average error of 371 GE

Table 2.3: Numerically estimated control overhead functions.

N	$\omega(N)$
32 bits	0.1518
64 bits	0.1186
96 bits	0.1192
128 bits	0.1103
$\|\text{residual}\|_2^2$	3.36E+06

F_{DP}	$\omega(F_{DP})$
0 - 2000 GE	0.1906
2000 - 4000 GE	0.1717
4000 - 6000 GE	0.1691
6000 - 8000 GE	0.1271
8000 - 10000 GE	0.1098
10000 - 12000 GE	0.0932
12000 - 14000 GE	0.0774
$\|\text{residual}\|_2^2$	1.38E+05

per design, the latter is greater by more than an order of magnitude (3.36E+06), meaning an error of around 1833 GE per estimation. This is also reasonable, as it appears to be more sensible that the amount of control logic depends more on the datapath area rather than on the length of registers.

Overall, using functional overheads such as these provide us with a more precise estimation of the total footprint area. For comparison with the plots discussed in previous section, Figs. 2.9 and 2.10 show the adjusted estimates for the training and test functions, respectively. Similarly, Fig. 2.11 shows the error distribution over test functions only. It is clear that the fit is now much more accurate (compare with Fig. 2.8), even though the new estimation cannot be regarded anymore as an upper bound for the total footprint area.

2.3 Conclusions

In this chapter, a study concerning the area of lightweight primitives used in lightweight cryptography has been presented. We have proposed a simple yet accurate procedure to estimate the footprint area of generic lightweight algorithms. We have argued that finding an accurate approximation is extremely hard, since it strongly depends on factors such as the architecture chosen by the designer, the manufacturing technology, the libraries used, the possibility of optimizing the footprint when combining several parts, etc. Despite this, the designer of algorithms for constrained environments (such as, for example, those related to cryptographic

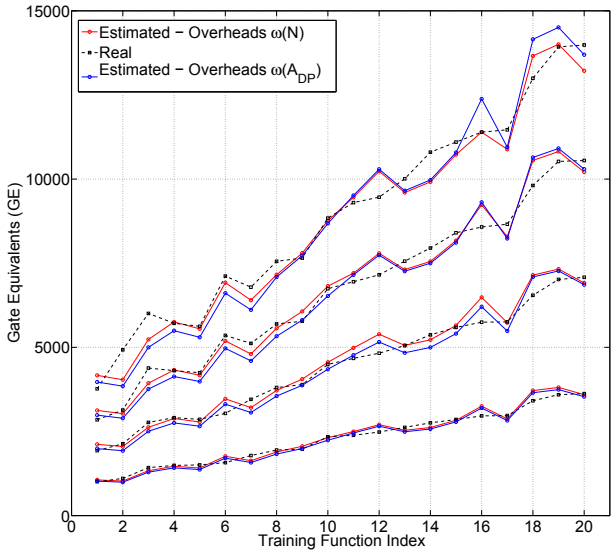


Figure 2.9: Real and estimated footprint area using adjusted control overheads: results on 80 training designs.

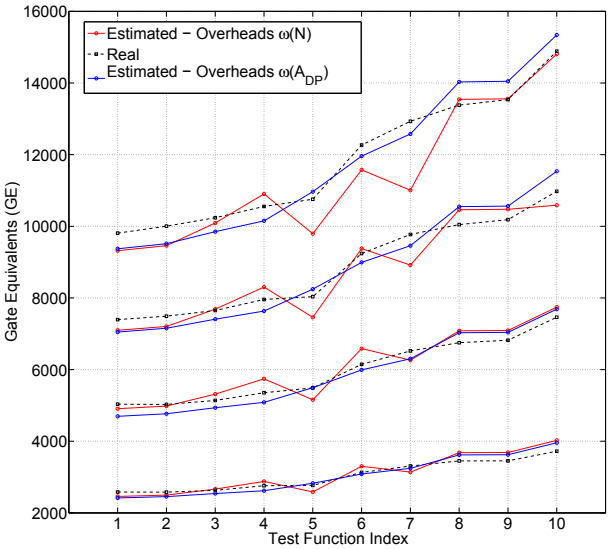


Figure 2.10: Real and estimated footprint area using adjusted control overheads: results on 40 test designs.

functions for RFID tags or sensor nodes) should count on some quantities to drive their choices. One major motivation for this work is to fill this gap by providing algorithm designers with a tool to estimate the cost, in terms of footprint area, of their constructions.

We believe our proposal will help in making some choices at the algorithmic level, even for designers without hardware design skills. Furthermore, it could also be applied to get preliminary comparisons among different proposals (lightweight primitives and more complex constructions such as security protocols) or, at least, to decide if they are simply too costly for certain operational environments.

The work presented in this chapter can be extended in a number of ways. One natural direction for future work is the inclusion of other commonly used elements in the F_{DP} estimator, such as for example S-boxes of non-linear filters. Similarly, we expect to test the proposed estimator against well-known lightweight cryptographic primitives and compare the predictions with reported experimental results. Finally, our focus in this work has been exclusively on the footprint area of ASIC implementations. It would be interesting to extend our estimates to include other prominent parameters, primarily throughput and power consumption, as these have also significant influence in design choices. Consider, for example, the RFID standards [47, 3], which states that a tag must support up to 1500 read attempts per second under ideal conditions. However, this rate can be five or ten times smaller (500-150 tags/sec) in real world environments [21]. Therefore, if the tag's operating frequency is set to 100 KHz, the number of clock cycles used per reading is upper-bounded by 670 (in fact, 500 clock cycles is an upper bound commonly assumed in previous works [94, 87]). The methodology discussed in this work can be easily extended to incorporate measures of throughput and power consumption. We expect to tackle this in future work.

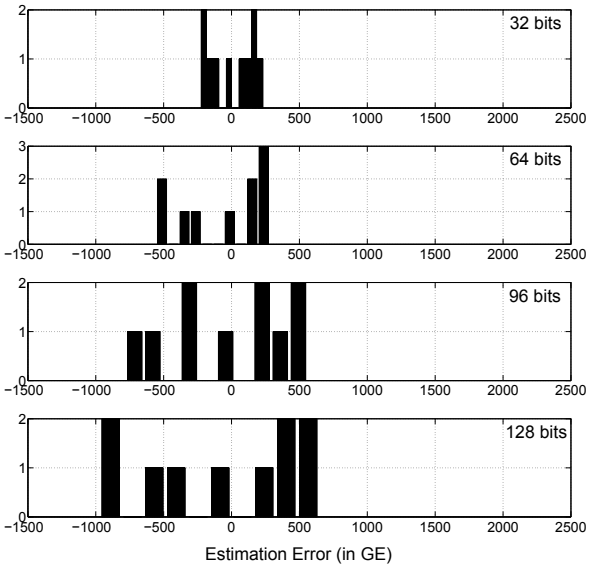


Figure 2.11: Distribution of gate count estimation errors using $\omega(F_{DP})$ on 40 test designs.

3

Pseudo-Random Number Generators

Pseudo-random number generators (PRNGs) are deterministic algorithms that generate an unbiased *random* output. The output sequence can be considered *random* in the sense that all the sequences have the same probability of appearance.

PRNGs are usually implemented in environments where True Random Number Generators are not feasible. Their operation mode is based on an unknown internal state, which is the *seed* to computes the next internal state. The output must have statistical properties indistinguishable from true random numbers.

These blocks are commonly used in cryptographic applications. As an example, the most well known application is the expansion key process, where using a initial key (*seed*), several subkeys are generated [95]. Other applications that use PRNGs are protocols [96], that use a shared secret and a Pseudo-Random Function (PRF) to protect the messages. One-time pad and nonces are also included among the common applications.

PRNGs serve as random number generators in current RFID technologies. They are usually implemented in tags and readers. The pseudorandomness offered by on board PRNGs is tipically used:

- To enhance the security of password-protected operations.
- In authentication protocols where the pseudo-random number is used to synchronize the reader and the tag [22].
- As an anti-collision mechanism for inventorying processes [47].

- To acknowledge other Gen2 specific operations (e.g., memory writing, decommission of tags, and self-destruction)[47].

PRNGs are, therefore, the crucial components that guarantee RFID security.

From a security perspective, The traditional definition of PRNGs involves a bunch of statistical tests, but this is insufficient for cryptographic purposes. A good PRNG for security applications should meet additionally the followings requirements [23]:

- The adversary cannot compute the internal state of the PRNG, even if many outputs of the PRNG have been observed.
- The adversary cannot compute the next output of the PRNG, even if many previous outputs of the PRNG have been observed.
- If the adversary can observe or even manipulate the input samples that are fed in the PRNG, but the internal state of the PRNG is not known, then the adversary cannot compute the next output and the next internal state of the PRNG.
- If the adversary has somehow learned the internal state of the PRNG, but the input samples that are fed in the PRNG cannot be observed, then the adversary cannot figure out the internal state of the PRNG after the re-keying operation.

Due to the fact that the standard EPC Class-1 Gen-2 requires the generation of a 16-bit pseudo-randon number, there is an increasing demand of secure PRNGs. These PRNGs should not only meet the requirements of EPC-C1G2 but should also offer interesting features (area,throughput, power consumption,etc) to be implemented in resource-constrained environments. In this chapter we present a set of PRNGs that meet with the standard EPC-C1G2 and are also suitable for being implemented in low-cost RFID tags.

The chapter is structured as follows: In the first section 3.1, a brief state of the art about PRNGs is presented, showing some general considerations. In section 3.2 the designed PRNGs are depicted and some details are also studied about the chosen architectures. In Section3.3 we have implemented two authentication protocols that

use the PRNGs that we have proposed. The hardware results obtained with up to 15 different designs are presented in Section 3.4. Finally, Section 3.5 concludes the chapter and summarizes our findings and contributions.

3.1 State of the art

3.1.1 PRNGs requirements for low-cost RFID tags

The security level offered by the EPC-C1G2 standard and other passive RFID tags is extremely low –in fact, almost inexistent. This is mainly due to the lack of computational resources on the tags, which prevents them from using standard security primitives and protocols. For example, it is commonly assumed (see, e.g.,[111]) that no more than 4000 Gate Equivalents (GE) can be devoted to security functions. As introduced before, a GE is the normalization commonly used for these applications. The GE value is obtained by dividing the whole area of the circuit by the area of a basic NAND gate. EPC-C1G2 tags support simultaneous read attempts up to 1500 tags/sec under ideal conditions. However, this rate can be five or ten times smaller (500-150 tags/sec) in real-world environments [21]. Therefore the number of clock cycles used per reading is upper-bounded by 670 clock cycles, assuming that the RFID chip operates at a clock frequency of 100 kHz. We take 500 clock cycles as reference value because this limit is less than the above mentioned value and has been used in previous works [87, 94]. Furthermore, they should not consume more than 10 μW, as low-cost tags are passive and, therefore, must harvest their power supply from the reader signal. (See, for example, [20] for an elaborate motivation on the need for low-power designs.) When these constraints are compared with the approximate 8120 GE [51, 102] required by a standard hash function like SHA-1 (which is an essential building block for most security protocols), it becomes clear the need for schemes that can provide some minimum security services while requiring as few resources as possible.

As aforementioned in previous chapters, EPC-C1G2 standard identifies three requirements that a PRNG must satisfy [47]:

- The probability that any RN16 drawn from the PRNG has value $RN16 = j$ for any j, shall be bounded by $0.8/2^{16} < P(RN16 = j) < 1.25/2^{16}$.

- For a tag population of up to 10,000 tags, the probability that any two or more tags simultaneously generate the same sequence of $RN16$ shall be less than 0.1%, regardless of when the tags are energized.

- An $RN16$ drawn from a tag's PRNG 10ms after the end of T_r (RF signal envelope rise time) shall not be predictable with a probability greater than 0.025% if the outcomes of prior draws from PRNG, performed under identical conditions, are known.

Due to the limitations imposed by the technology requirements and the EPC-C1G2 standard, it is a challenging problem to design a PRNG suitable for low-cost RFID tags.

3.1.2 PRNG evaluation

The PRNG output must be indistinguishable from a random number. The statistical properties of the output are usually examined using battery tests and then compared to the statistical properties of truly random sequences. These tests do not guarantee, when successfully passed, that a given generator is useful for all kind of applications. As aforesaid, the security analysis of PRNGs in a cryptographic context is not restricted to a statistical analysis of the output of the generator, the algorithm itself must be cryptographically analyzed in order to avoid some weaknesses as linearity.

The most well-known tests used to evaluate PRNGs are:

- ENT: ENT is a statistical test designed to evaluate PRNGs but it is also used to test random streams [142]. The tests carried out by this suite are: Entropy, Chi-square Test, Arithmetic Mean, Monte Carlo Value for Pi and Serial Correlation Coefficient.

- NIST Test Suite for random Number Generators: this statistical package has been developed by NIST [114]. The package consists of 15 tests that evaluate the distribution of long binary sequences. The tests are: Frequency (Monobits) Test, Test for Frequency within a Block, Runs Test, Test for the Longest Run of Ones in a Block, Random Binary Matrix Rank Test, Random Binary Matrix Rank Test, Non-overlapping (Aperiodic) Template Matching Test, Overlapping

(Periodic) Template Matching Test, Maurer's Universal Statistical Test, Linear Complexity Test, Serial Test, Approximate Entropy Test, Cumulative Sum (Cusum) Test, Random Excursions Test and Random Excursions Variant Test.

- Diehard: It is a test battery consisting of 15 test developed by Georges Marsaglia in 1996 [91]. It is considered one of the most tough test batteries, some generators that pass NIST test fail DIEHARD. The tests that compound the battery are: birthday spacings , overlapping permutations, ranks of 31x31 and 32x32 matrices, ranks of 6x8 matrices, monkey tests on 20 bit Words, monkey tests OPSO, OQSO, DNA, count the $1's$ in a stream of bytes, count the $1's$ in specific bytes, parking lot, minimum distance, random spheres, squeeze, overlapping sums, runs, and craps. In 2002, Marsaglia [92] proposed a reduction of the tests. Only 3 tests were selected for this lightweight suite and integer numbers reduced to only 32 bits to evaluate the randomness of the sequence. The new 3 tests are: The gcd test, based on Euclid's algorithm for computing the gcd of two random 32-bits integers. The Gorilla test, a stronger version of Monkey test presented in Diehard. The Birthday Spacing test, a stronger version of iterated birthday spacing test presented in Diehard.

Besides statistical tests as a way of randomness evaluation, there are other methods related to the verification of EPC-C1G2 functionalities that can be carried out using a simulation platform. For example, the IAIK UHF Demo Tag [101] is a programmable device intended for developing new commands or functionalities to the EPC Gen2 standard. It allows, moreover, to verify the new functionality using compliant EPC Gen2 readers, by modifying the code inserted into the Demo Tag. Thus, new developments can be implemented and tested in real environments [121].

3.1.3 Known PRNGs

PRNGs have been studied and used for some decades. There are several PRNGs well-known in the literature and some secure proposals from different authors. Among the most well-known PRNGs can be highlighted the following ones:

- **Linear Congruential Generator (LCG):** LCGs are simple pseudo-random number generators usually defined by the following equation:

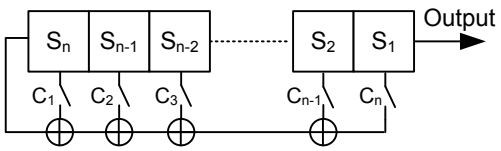


Figure 3.1: Linear Feedback Shift Register scheme

$$X_{n+1} = a \cdot X_n + c \ (mod \ m)$$

where X_0 is the seed and a, c and m are constants. If these constants are selected carefully, the LCG will be a maximal period generator. LCGs are very popular due to their straightforward implementation and good statistical characteristics, but are insecure from a cryptographic point of view. The underlying problem of LCGs is that X_{n+1} and X_n are not independent. An attacker could predict the entire sequence after eavesdropping some outputs (even if a, c and m are unknown). LCGs are not used for cryptographic applications for this reason.

- **Linear Feedback Shift Register (LFSR):** LFSRs are generators that consist of a shift register and XOR gates (see Fig.3.1). The LFSR input bit is a linear function of its previous state. The feedback polynomial function $(C(x) = 1 + c_1x^1 + c_2x^2 + ... + c_nx^n)$ establishes the register positions that are XORed. If the feedback function is primitive, the LFSR will have maximal length. The main weakness that present LFSRs is the linearity.

- **Blum-Blum-Shub Generator [17]:** Blum-Blum-Shub security depends on an integer factorization that is generally assumed to be intractable. A pseudo-random bit sequence $(z_1, z_2, ...z_l)$ of length l is generated as described below:

  1 **Setup**

  1.1 Generate two large secret random (and distinct) primes p and q each congruent to 3 modulo 4, and compute $n = p \cdot q$.

  1.2 Initialize the seed s randomly in the interval $[1, n-1]$ such that $gcd(s, n) = 1$ and compute $x_0 = s^2 \ mod$ N.

  2 **Random Bit Generation** For i from 1 to l:

  2.1 $x_i = x_{i-1}^2 mod n$

  2.2 z_i =least significan bit of x_i

The main problem of this generator is that it needs a lot of resources for the implementation due to the computational complexity.

The previous PRNGs can not be used in low-cost RFID tags because of the lack of security (LCG and LFSR) or the resources used in its implementation (BBS). There are many others PRNGs in the literature, for example those based on the theory of chaos. But these PRNGs do not accomplish the EPC-C1G2 requirements, or they are not suitable for low-cost RFID tags.

In the last decade, several schemes have been reported. Among the PRNG proposals oriented to low-cost RFID tags stand out the following ones:

- **LAMED:** Peris-Lopez et al. present in [105] a deterministic algorithm that relies on the use of 32-bit keys and pre-established initial states. The set of functions mainly consists of bit rotations, bitwise operations, and modular algebra, building a 32-bit PRNG. The authors also propose an alternative 16-bit version of their PRNG for EPC Gen2 compatibility. To reduce the output length from 32 to 16 bits, Peris et al. divide the 32-bit output in two halves and XOR them to obtain the 16-bit output sequence. Experimental results regarding area, throughput and power consumption were not presented.

- **Che et al.:** Che et al. describe in [28] a hybrid approach that combines the use of Linear Feedback Shift Registers (LFSR) and a TRNG to build random sequences. A theoretical attack to this PRNG was presented in [121].

- **J. Melia-Segui et al:** In [94] J. Melia-Segui et al. handled the inherent linearity of LFSRs by means of a multiple-polynomial approach. Several feedback functions were implemented. The selection of each primitive polynomial for every cycle is performed by the true random data source and a decoder. The authors present a secure PRNG design suitable to the current EPC Gen2 technology, providing evidences of statistical and hardware compatibility. The main problem is that the PRNG security relies on a TRNG.

Based on the necessity of embedded PRNGs in tags, and taking into account the lightweight designs reported in the literature, we have proposed two lightweight PRNGs that meet the EPC-C1G2 requirements, and are suitable for low-cost RFID tags.

3.2 AKARI-X

3.2.1 Design and evaluation

Achieving EPC-C1G2 with a lightweight design supposes a challenge, as the lack of resources in the tag impose severe constraints on the type and amount of operations that can be included. Over the last years, several authors have proposed various designs appropriate for resource-constrained devices such as low-cost RFID tags (see, e.g., [27, 94]).

In this context, the concept of T-function introduced by Klimov and Shamir [74, 75] results very interesting. All bitwise operations (e.g. bitwise XOR ($a \oplus b$), OR($a \vee b$) and AND ($a \wedge b$)) and most of the modern machine operations (e.g. addition ($a + bmod 2^m$) and multiplication ($a \cdot bmod 2^m$)) are T-functions and their composition are also T-functions.

In particular, the mapping $x \rightarrow x + (x \cdot 2 \vee C) (\text{mod } 2^m)$ is very interesting. For any m, it is a permutation with a single cycle of length 2^m if both the least significant bit and the third significant bit in the constant C are 1. Furthermore, the output provided by this permutation looks like a random variable. However, this function is not cryptographically secure. More precisely, an attacker can exploit the fact that when a T-function is executed there is not propagation of information from left to right, which facilitates its cryptanalysis. Nonetheless, these T-functions can be mixed with a non-linear function to increase its security.

Using the information about the desirable cryptographic features of T-functions and the study performed in chapter 2 about the footprint area of lightweight elements, two lightweight PRNGs called AKARI-1 and AKARI-2 have been designed.

To overcome the above mentioned drawback of T-functions (no propagation of information from left to right) and guarantee a high degree of diffusion, we have included in our designs non-linear filters. Several candidates have been generated using genetic programming. The possible non-linear filters are obtained through evolving compositions of extremely light operands in terms of computation and hardware demands (multiplication was excluded because of the results presented in chapter 2). It is noteworthy that the non-linearity of each candidate was measured using the Avalanche effect concept. This property tries, to some extend, to reflect the

AKARI-1

1. $x_0 = x_0 + ((x_0 \cdot 2) \vee 5)$
2. $x_1 = x_1 + ((x_1 \cdot 2) \vee 13)$
3. $z = x_0$
4. **for** $r = 0$ to 63 **do**
5. $\quad z = (z \gg 1) + (z \ll 1) + z + x_1$
6. **Return** least significant $m/2$ bits of z

AKARI-2

1. $x_0 = x_0 + ((x_0 \cdot 2) \vee 5)$
2. $x_1 = x_1 + ((x_1 \cdot 2) \vee 13)$
3. $z = x_0 \oplus x_1$
4. **for** $r = 0$ to 24 **do**
5. $\quad z = (z \ll 1) + ((z + \texttt{0x56AB0A}) > 1)$
6. $y = x_1 \oplus z$
7. **for** $r = 0$ to 24 **do**
8. $\quad y = (y \gg 1) + (y \ll 1) + y + \texttt{0x72A4FB}$
9. **Return** least significant $m/2$ bits of $z \oplus y$

Figure 3.2: Pseudorandom Number Generators AKARI-1 and AKARI-2.

intuitive idea of high non-linearity. More specifically, the avalanche effect is evident if, when an input is changed slightly, the output changes significantly. Further information about the methodology used to obtain non-linear filters can be found in [64].

A description in pseudocode of the two PRNGs is given in Fig. 3.2, where ($\gg$) and ($\ll$) symbolize right and left circular shift, respectively.

Analysing the proposed alternatives, AKARI designs are based on iterating a simple function a given number of rounds. AKARI-1 consists of a initialization phase that is usually precomputed in the reader (1,2 and 3) and only one filter function which is iterated a relatively high number of times ($r = 64$). This function consists of simple additions ($a + b mod 2^m$) and right and left circular shifts. Oppossedly, AKARI-2 consists of an initialization phase(1,2 and 3) but employs two filter functions. The two filter functions are mixed, facilitating the reduction of the number of iterations in the loop $r = 24$. These two filters include lightweight operations, circular shifts and constants. It is important to note that in both cases, the algorithm uses variables of m bits and outputs numbers of $m/2$ bits.

Table 3.1: Evaluation of the quality of AKARI-1 and AKARI-2 ($m = 32$) against several randomness tests.

Battery	Test	AKARI-1	AKARI-2
ENT	Entropy	8.000000	8.000000
	Compression Rate	0	0
	χ^2 Statistic	259.09 (41.70%)	250.99 (55.93%)
	Arithmetic Mean	127.4976	127.5031
	MonteCarlo π estimation	3.141447036 (error 0.00%)	3.141512474 (error 0.00%)
	Serial correlation coefficient	-0.000026	3.141512474 (0.000013
Diehard	Overall p-value	0.352645	0.551129
NIST	All	Pass	Pass

The statistical quality of the output sequence generated by AKARI-1 and AKARI-2 was analyzed using three batteries of statistical randomness tests: ENT [142], DIEHARD [91] and NIST [114]. The results obtained in these tests are depicted in Table 3.1

3.2.2 Hardware architectures

We next describe several architectures for the implementation of both AKARI PRNGs. The main goal of the designs depicted in Fig. 3.2 is meeting the various technology requirements for low-cots RFID tags, namely using less than 4000 GE, taking less than 500 clock cycles in the generation of a random number, and consuming less than 10 μW [111]. At the same time, we try to maximize the security level, which is directly linked to the bit-length of the generated random numbers.

With the proposed architectures, we try to maximize the throughput or minimize the area while the above mentioned requirements are fulfilled. A reduction in the footprint area can be exploited to add additional bits for the operations (more security) or alternatively it can be used just to reduce the cost of silicon area.

To maximize the throughput, the best strategy is trying that all the operations are computed in only one clock cycle, as the clock frequency is very low at 100 KHz, and there is plenty of room for a long critical path. With this strategy, the number of clock cycles necessary to generate a Pseudo-random number are reduced to the minimum.

On the other hand, the strategy adopted to reduce the area is related to the lightweightness of the operations. As shown in the previous section, both PRNGs mainly use simple bitwise operations. Even though these have a reduced cost in terms of area, they can be implemented in different ways and it is unclear which one will best fit the requirements given above. Taking into account the study carried out in chapter 2, we can summarize that adders are the most *expensive* operation, in terms of area, involved in these algorithms. We concluded that the best way to improve the area cost is to reduce the size of the adder. However, this strategy incurs in a penalty in throughput, since each addition takes several clock cycles. The limit of this area reduction is established by the number of clock cycles used. As aforesaid, some works assume that an EPC-C1G2 protocol should take less than 500 clock cycles for each protocol run. For a 100 kHz frequency, this means one authentication each 5 milliseconds or, equivalently, 200 protocol runs per second.

We have proposed several implementation architectures for each PRNG in order to explore the trade-off between area and throughput while the EPC-C1G2 requirements are achieved.

3.2.2.1 AKARI-1 Architectures

With the first architecture (AKARI-1A) we attempt to minimize the number of clock cycles required to generate an output. Each operation is executed in only one cycle whenever this is feasible. To achieve this, different m-bit operation blocks are used, m being the bit length of the variables, and the control of inputs/outputs to/from each block is implemented through a Finite State Machine (FSM).

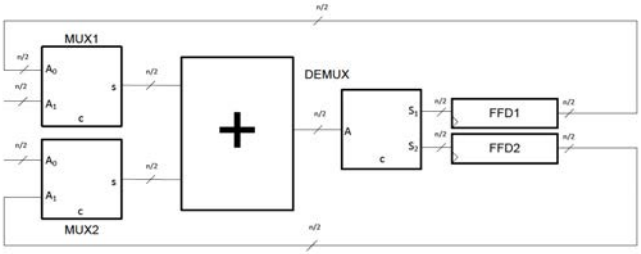
In the second architecture (AKARI-1B) we seek to reduce the overall chip area by reducing the area occupied by the adder. More precisely, we use an adder with half the number of bits ($m/2$) plus the necessary control implemented by an FSM. With this approach, the circuit needs more clock cycles because each sum takes now 2 cycles rather than just 1. Besides, it is now necessary to add some additional

logic (multiplexers and demultiplexer) to select which input is used at each cycle (see Fig.3.3(a)). A priori, it is unclear whether the improvement in area given by a reduced adder will or will not compensate for the area required by additional logic. This will be discussed later when analyzing the implemented circuits.

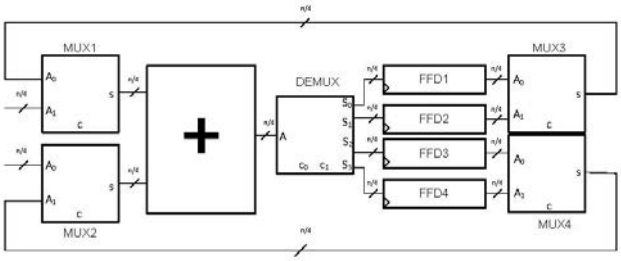
It is not possible to go further with this reduction strategy in AKARI-1 PRNG (using an adder of $(m/4)$) because the number of clock cycles used exceeds the limit imposed by the standard.

3.2.2.2 AKARI-2 Architectures

We have explored three different architectures for AKARI-2. The first two (AKARI-2A and AKARI-2B, respectively) are identical to those developed for AKARI-1; i.e., AKARI-2A uses m-bit adders and takes 1 cycle, while AKARI-2B uses $m/2$-bit adders and takes 2 cycles. To further explore the trade-off between area and throughput, the third approach (AKARI-2C) goes one step further and uses $m/4$-bit adders with additional support logic. In detail Fig. 3.3(b) we show the $m/4$ adder plus the logic control (multiplexers and D flip-flops) that is needed to implement this approach.



(a) with half bit length $(m/2)$



(b) Adder with quarter bit length $(m/4)$

Figure 3.3: Half $(m/2)$ and quarter $(m/4)$ adders and auxiliary logic and registers.

For the same reason as in AKARI-1, it is not possible to go further with the reduction strategy.

We have presented several architectures for both PRNGs in this section. However, PRNGs are typically used along with authentication protocols that take advantage of the freshness generated by PRNGs to increase their security. In the next section are presented two authentication protocols compliant with EPC-C1G2 that require a PRNG.

3.3 RFID authentication Protocols

To provide security in RFID communications, authentication protocols turn out to be very effective. In [34] Chien proposed the following classification about the hardware complexity of the different kind of protocols used in RFID: 1) Full-fledged tags support on-board conventional cryptography like symmetric encryption, cryptographic one-way functions and even public key cryptography; 2) Simple tags support random number generators and one-way hash functions; 3) Lightweight tags support a random number generator and simple functions, such as for example a Cyclic Redundancy Code (CRC) checksum, but not a cryptographic hash function. Ultra-lightweight tags can only compute simple bitwise operations, like XOR, AND, OR, etc.

A few security lightweight protocols are proposed in the literature aimed at meeting the technology requirements set by the EPC-C1G2 standard [47]. In these proposals, Pseudorandom Number Generators (PRNG) are often used to provide freshness to the generated messages and avoid some attacks. According to EPC-C1G2, tags should be able to generate 16-bit pseudorandom numbers (RN16), and store temporarily at least two of these values. The system mainly comprises interrogators (readers) and labels (tags). An interrogator manages tag populations using three basic operations: 1) *Select* - the operation used to choose a tag population; 2) *Inventory* - to identify tags; and 3) *Access* - the operation used for reading from and/or writing to a tag.

We next describe two representative ultra-lightweight protocols based on PRNGs developed to conform with the EPC-C1G2 specification.

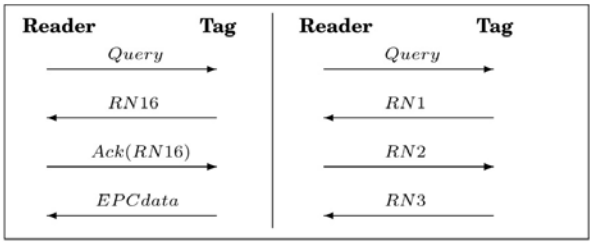


Figure 3.4: The 4-pass EPCGen2 inventory (left) and Burmester-Munilla inventory (right) [22].

3.3.1 Burmester-Munilla Protocol

Burmester and Munilla [22] proposed a lightweight mutual authentication RFID protocol that supports session unlinkability and forward and backward security. In this protocol, each tag shares with the server a synchronized PRNG (same algorithm, key and seed). Tag and server are mutually authenticated by exchanging either three or five consecutive numbers from the PRNG. Moreover the PRNG can be refreshed when there is suspicion that the state of the PRNG may be compromised.

The original EPC-C1G2 protocol has four passes for identification, which involve the exchange of the following messages: a query, a random number $RN16$, an acknowledgement $ACK(RN16)$ and the EPC data. As shown in Fig. 3.4, in [22] the authors replace these values by three random numbers ($RN1$, $RN2$ and $RN3$) in the so-called optimistic case. If $RN1$ was used previously (a flag called alarm is ON), then two more nonces ($RN4$ and $RN5$) have to be exchanged.

3.3.2 Chien-Huang Protocol

Chien and Huang presented in [33] a new authentication protocol based on Li et al.'s scheme [150]. The authors showed that Li et al.'s scheme is vulnerable against replay attacks and attempted to improve its security level while preserving its lightweight properties. The security of Chien-Huang protocol is based on a synchronized PRNG shared between the tag and the reader. The scheme supports mutual authentication and the authors claim that it provides security against replay, traceability and DOS attacks.

A short description of the protocol is given below, but the reader is referred to the original paper for further details. Each tag stores an l-bit secret key x, an l-bit

Table 3.2: Security Properties

	Burmester-Munilla	Chien-Huang
Mutual authentication	0	0
Replay Attacks	0	0
DOS Attacks	0	0
Active Attacks	0	△
Forward secrecy	0	0
No traceability	0	△

×- no satisfied, 0 - satisfied, △ - partially satisfied

secure identity SID, and an l-bit index-pseudonym IDS. Six values are stored in the database for each tag: a secure identity SID, the current index-pseudonym IDS_{new}, the old index-pseudonym IDS_{old}, the current key x_{new}, the old key x_{old}, and a $flag$ bit that is used to check whether the tag and the database are synchronized or not. Three operators are used: 1) a PRNG $g()$; 2) $rotate(p, w)$, which left rotates the operand p w positions; and 3) $Left(s)$ and $Right(s)$, which symbolize the left/right half of s respectively. Fig. 3.5 depicts the exchanged messages in this scheme.

Table 3.2 presents a brief security comparison between the two chosen protocols. Both protocols provide mutual authentication and Burmester-Munilla scheme seems to be more robust offering protection against traceability and active attacks.

3.3.3 Design architectures for RFID identification protocols

In this section, we present the designs made for the two EPC-C1G2 protocols described above and the hardware architectures chosen, including memory, computational and control logic. Both schemes rely on the use of a sufficiently good PRNG, but the particular choice is left to implementers. Given that such a component is critical to guarantee that the resulting circuit will fit a low-cost RFID tag, we have explored the two AKARI designs presented above.

Most PRNG-based EPC-C1G2 protocols follow a similar working scheme. We have designed an architecture for a generic EPC-C1G2 protocol (see Fig. 3.6) and then particularized it for each implemented protocol. This architecture includes four main blocks:

- Register Block: This encompasses all the registers needed to store intermediate computations and long-term values. For example, in the Burmester-Munilla

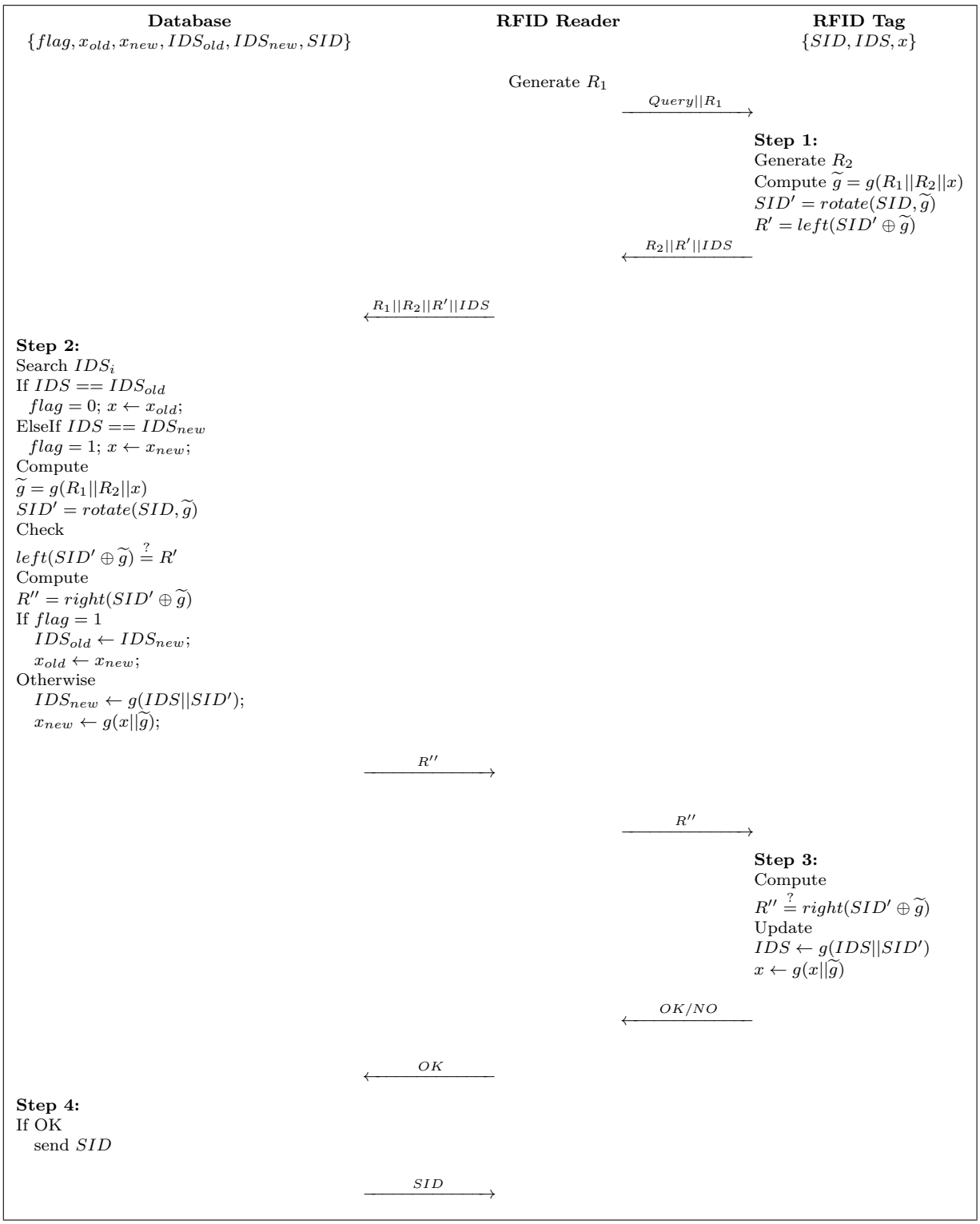


Figure 3.5: Chien and Huang lightweight RFID authentication protocol [33].

protocol it contains the registers that store RN_1, RN_2, the state ($g_{tag}(state)$) plus the refresh key K (we can discard the 1-bit flag *cnt* as its cost is negligible). Likewise, in the Chien-Huang protocol the block stores the internal values (SID, IDS and x) and the nonce R_1 received from the reader.

- PRNG Block: It implements the chosen pseudorandom number generator.
- Timer Block: It controls the maximum waiting time for each message exchange during a protocol run, indicating if the current execution must be aborted if the other party does not reply.
- FSM: The interaction among the different block elements during a protocol run is controled by a protocol-specific FSM. It also implements other details of each scheme. For example, in the Burmester-Munilla protocol it checks the alarm signal and selects the different operation modes (optimistic case or worst case).

As far as optimization is concerned, most efforts are concentrated on the PRNG block. The remaining modules are mainly composed of basic blocks and there is not much room for optimization. Despite this, in some cases we were able to reduce area by reusing some logic components from the PRNG into the protocol FSM.

3.4 Circuit Synthesis and Results

In this section, we report and discuss the main findings obtained for the different architectures described above. Firstly, we implemented the 3+2 choices for the PRNG using 6 different bit lengths (8, 16, 32, 64, 128 and 256 bits), resulting in 30 different designs. Each protocol was then implemented with 15 of them (32, 64 and 128 bits), as the remaining alternatives are inadequate for the EPC-C1G2 standard.

3.4.1 Experimental Setting

The synthesis of the various implementation alternatives discussed above was done with the Synopsys software. The hardware was described in VHDL language, using a structural style. The UMC library Faraday 90nm has been used. The choice of

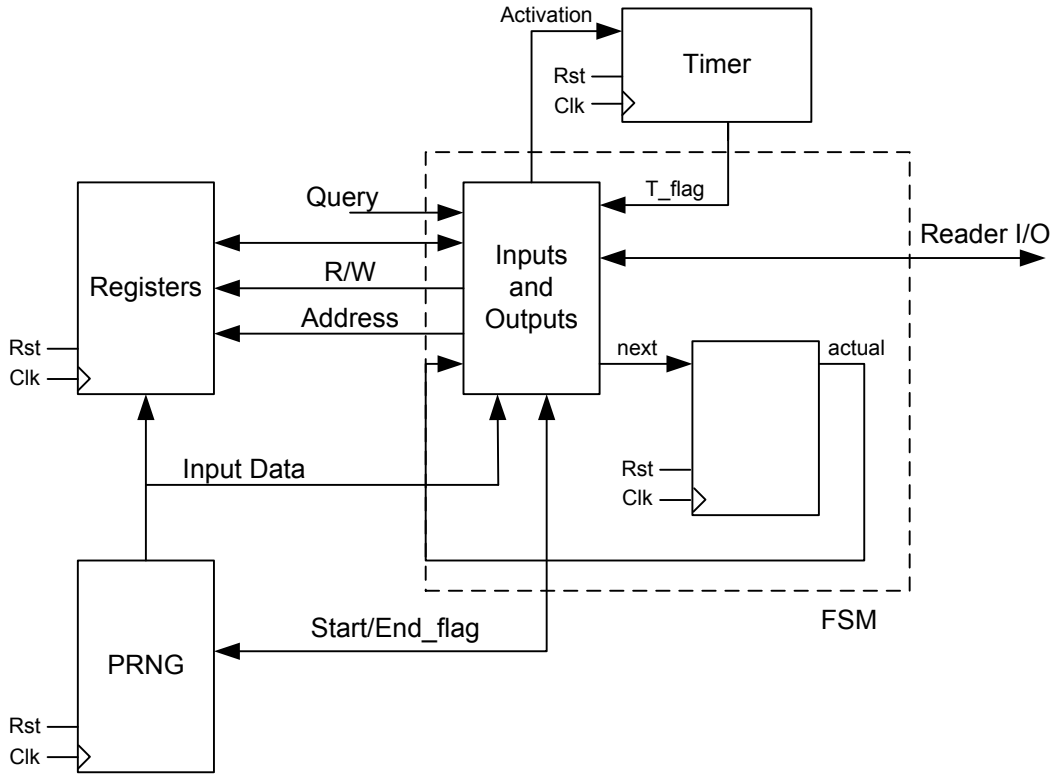


Figure 3.6: Hardware architecture for a generic EPC-C1G2 protocol.

this library is motivated by the fact that it provides information at the cell level, giving access to very valuable information that is generally unavailable when using generic libraries. In particular, this library provides detailed information of the standard cells' layout, which allows us to have a good estimation of the area and power consumption of the circuit. Overall, this guarantees that the results here presented are similar to those that would be obtained in a manufactured circuit. Although the final figures may suffer slight variations, these are reasonably bounded and do not have a significant impact in the results here reported.

All the tests have been performed with an operating frequency of 100 kHz for the clock signal of the integrated circuit, which is typically used in RFID systems, and a power supply set to 1.2 V. During our experimentation, it was found that the best results were obtained using a medium effort in map, area and power consumption.

Three metrics were used to analyze the proposed implementations:

- **Area:** The full area occupied by the circuit, both in μm^2 and in Gate Equivalents (GE), is presented. As aforesaid, the GE is obtained by dividing the whole circuit area by the area of a basic NAND logic gate; this result is completely independent of the particular technology used.

- **Power consumption:** An estimation of the power consumption is provided. Such a quantity heavily depends on the chosen technology.

- **Throughput:** The circuit throughput ($Kbps$), which measures how fast outputs are generated, is presented.

3.4.2 PRNG Results

3.4.2.1 AKARI-1 Results

Table 3.3 summarizes the four synthesis metrics identified above (circuit area, GE, power consumption and throughput) for each PRNG architecture and using six different output bit lengths. Some general conclusions can be drawn from these results:

Table 3.3: Hardware Analysis of AKARI-1 PRNG.

	8 bits	16 bits	32 bits	64 bits	128 bits	256 bits
Area (μm^2:)						
AKARI-1A	1494	3191	6209	12224	24340	48563
AKARI-1B	1643	2892	5484	10669	20912	41406
Gate Equivalents (GE):						
AKARI-1A	476	1018	1980	3898	7762	15486
AKARI-1B	524	922	1749	3402	6669	13204
Power (nW):						
AKARI-1A	47.35	89.95	173.8	343.2	712.17	1410
AKARI-1B	54.61	95.71	182.32	350.36	710.20	1460
Throughput (Kbps):						
AKARI-1A	24.24	48.48	96.96	193.92	387.84	775.68
AKARI-1B	3.55	7.11	14.22	28.44	56.88	113.77

1. There is a clear trade-off between area and throughput. For example, for a 256-bit architecture, it can chosen between generating a number with a minimal number of clock cycles (66 in AKARI-1A) or around a 15% improvement in area (AKARI-1B) with a serious penalty in throughput.

2. The improvement in area becomes more noticeable for architectures with larger bit lengths. This is reasonable, as the impact of the additional logic required by the sequential adder is increasingly less relevant.

3. Differences in power consumption are not significant. However, we emphasize that this strongly depends on the fabrication technology employed and, therefore, these figures have to be taken with care.

Circuit area is a severe restriction in lightweight cryptography. It is commonly assumed that a maximum of 4000 GE can be devoted for security functions. In Fig. 3.7 we can observe that the area increases linearly with the number of bits. For a maximum of 4K GE, we estimate that the output bit-length is upper bounded by 65 bits (AKARI-1A) and 75 bits (AKARI-1B), respectively. Fig. 3.8 shows the area, power and clock cycles for a 64-bit implementation of the two architectures. AKARI1-A fulfills the area requirements, and the required number of clock cycles is also quite below the limit ($66 \ll 500$). AKARI-1B shows a different trade-off between area and throughput, but the number of clock cycles is still below the 500 limit (450).

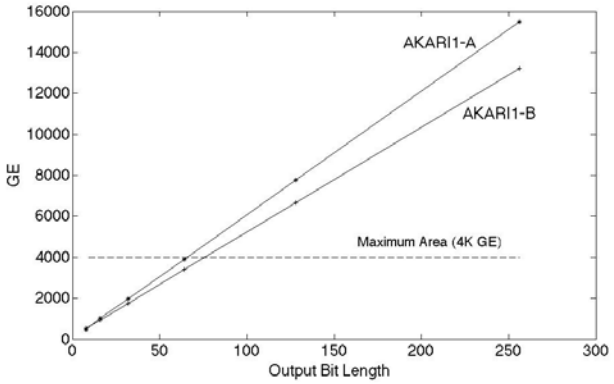


Figure 3.7: Area analysis of AKARI-1 PRNG (Gates Equivalents).

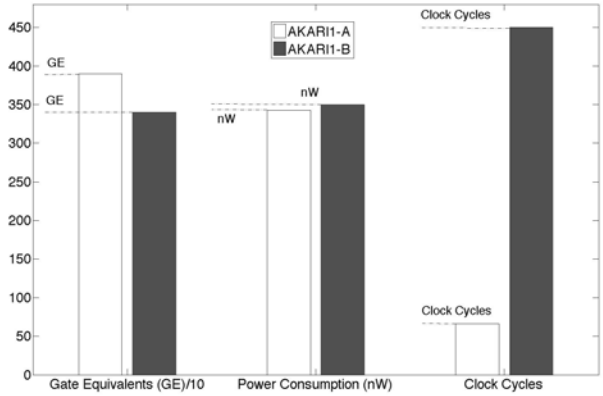


Figure 3.8: Implementation results of AKARI-1 PRNG (64-bit architecture).

3.4.2.2 AKARI-2 Results

The synthesis results for the three implementations of AKARI-2 follow a pattern similar to that observed for AKARI-1 (see Table 3.4). The first proposed architecture (AKARI-2A) optimizes speed (51 clock cycles), while the third one (AKARI-2C) optimizes the area at the expense of a lower throughput. AKARI-2B sits somewhere in between, but only for bit lengths greater than 16 bits. Fig.3.9 shows the area occupied by the three architectures as a function of the bit length. The larger values that meet the 4K GE requirement are 34.2 (AKARI-2A), 40.4 (AKARI-2B) and 42.8 bits (AKARI-2C), respectively. Therefore, a 32-bit output seems a reasonable choice if AKARI-2 is to be used in a protocol. Fig. 3.10 summarizes the performance characteristics for this bit length. The three architectures consume roughly the same

Table 3.4: Hardware Analysis of AKARI-2 PRNG.

	8 bits	16 bits	32 bits	64 bits	128 bits	256 bits
Area (μm^2:)						
AKARI-2A	2582	5837	11740	23462	46955	93257
AKARI-2B	2794	5173	10014	19656	38641	76910
AKARI-2C	2831	5081	9534	18421	36241	71579
Gate Equivalents (GE):						
AKARI-2A	824	1.861	3.744	7.482	14.973	29.738
AKARI-2B	891	1.650	3.193	6.268	12.322	24.525
AKARI-2C	903	1.620	3.040	5.874	11.557	22.825
Power (nW):						
AKARI-2A	57.38	109.88	216.06	439.37	902.49	1790.00
AKARI-2B	76.95	135.81	255.28	522.04	1070.00	2300.00
AKARI-2C	72.33	126.02	231.25	454.34	924.81	1810.00
Throughput (Kbps):						
AKARI-2A	15.68	31.37	62.74	125.49	250.90	501.96
AKARI-2B	2.75	5.50	11.03	22.06	44.13	88.27
AKARI-2C	1.50	3.01	6.03	12.07	24.15	48.30

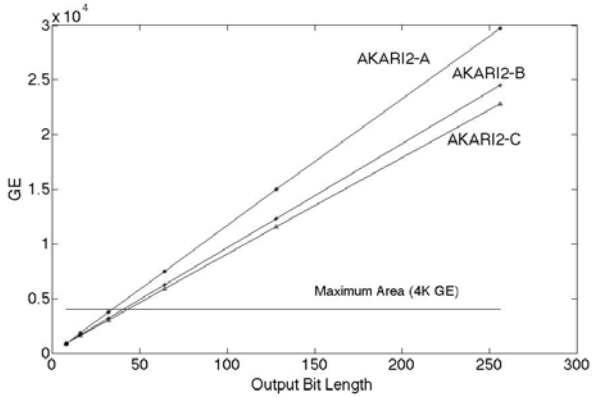


Figure 3.9: Area analysis of AKARI-1 PRNG.

power. There is no significant difference in area either, as the benefits of using smaller adders (AKARI-2B and AKARI-2C) do not become apparent for such a small bit length. This, however, seriously affect throughput, which drops dramatically for AKARI-2C. Such bad performance of AKARI-2C in terms of throughput does not come as a surprise, as the use of $m/4$-adders implies that each sum requires four clock cycles, rather than just 1 and 2 in AKARI-2A and AKARI-2B, respectively.

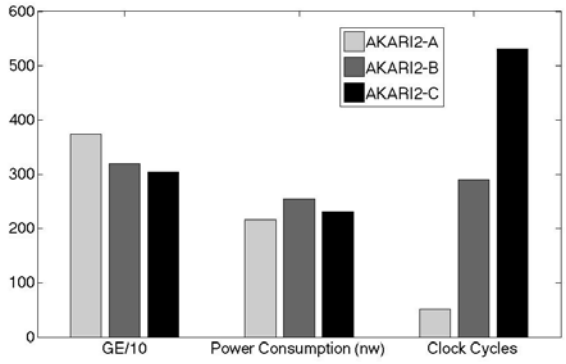


Figure 3.10: Implementation results of AKARI-2 PRNG (32-bit architecture).

3.4.3 Protocol Results

We now discuss the synthesis metrics obtained for the two protocols described in Section 3.3. Each protocol has been implemented using the 5 PRNGs architectures for 3 different bit lengths: 32, 64 and 128 bits.

3.4.3.1 Burmester-Munilla Protocol

Table 3.5 shows the footprint area, power consumption and maximum number of authentications per second for Burmester-Munilla protocol. The area remains below (or close to) 4K GE for the 5 PRNG implementations for a 32-bit architecture. For larger bit-lengths, the area grows significantly, particularly if AKARI-2A is used. Further investigation reveals that between 60% and 90% of the chip is occupied by the PRNG. To better illustrate this, Fig.3.11 shows the total chip area and the fraction corresponding to the PRNG. This gives an interesting insight for protocol designers.

As for power consumption, an upper bound of 10 μW is commonly assumed as the maximum power that a passive RFID tag can consume. Our implementations are well below that limit, even for larger bit lengths such as 128 bits. The differences observed between the power consumption of an isolated PRNG (see Tables 3.3 and 3.4) and the entire protocol are due to several facts: several random numbers are generated in each protocol execution, the chip is energized for a longer period of time, and the extra protocol circuitry adds some nano-watts.

Table 3.5: Hardware analysis of Burmester-Munilla EPC-C1G2 protocol.

Gate Equivalents (GE):

PRNG	32 bits	64 bits	128 bits
AKARI-1A	2666	5184	11382
AKARI-1B	2557	4833	9227
AKARI-2A	4026	8010	18282
AKARI-2B	3427	6766	13037
AKARI-2C	3519	6659	12723

Power (nW):

PRNG	32 bits	64 bits	128 bits
AKARI-1A	311	614	1250
AKARI-1B	288	530	1051
AKARI-2A	338	643	1266
AKARI-2B	326	622	1239
AKARI-2C	321	610	1162

Authentications/second

PRNG	Best case	Worst case
AKARI-1A	245	147
AKARI-1B	37	22
AKARI-2A	314	188
AKARI-2B	57	34
AKARI-2C	31	18

We have also calculated the number of authentications per second of each protocol implementation. As described in the original protocol, the authentication could involve the exchange of 3 (optimistic case) or 5 (worst case) random numbers. This makes a difference, as the tags must wait for the reader to generate further numbers, and then receive and process them. Throughput plays a fundamental role here, as shown in 3.5. In both cases, different PRNG architectures yield substantially different throughput results.

3.4.3.2 Chien-Huang Protocol

Implementation results for Chien-Huang protocol are quite similar to those discussed for the first protocol (see Table 3.6). Again, a 32-bit architecture seems the fittest choice for a low-cost RFID tag in terms of area, as larger bit lengths result in a bigger

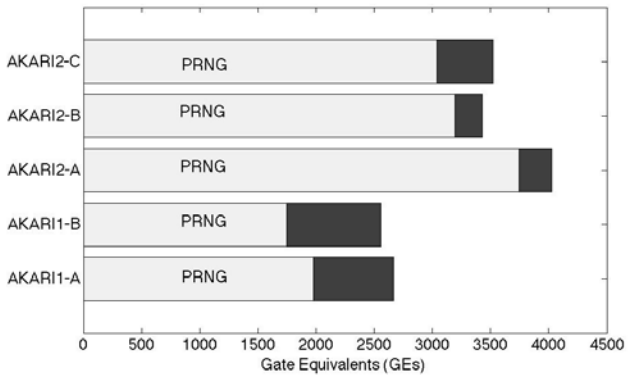


Figure 3.11: Area analysis of Burmester-Munilla protocol (32-bit architecture).

footprint. Fig.3.12 shows the fraction of the total chip area occupied by the PRNG. As in the case of Burmester-Mulilla protocol, the PRNG accounts for more than 60% of the total area, reaching an 80% for AKARI-2. In any case, all footprints are quite power efficient and there are no significant differences between both protocols.

According to our implementations, Chieng-Huang protocol is clearly faster than Burmester-Munilla, with a difference of more than 100 authentications per second for the fastest versions of both schemes. This is reasonable, as Chieng-Huang involves the generation of just 1 random number, while Burmester-Munilla requires 3 or 5.

3.4.4 Impact of EPC-C1G2 module in RFID tags

As an example, in Fig. 3.13 we show a general architecture for a battery-less wireless sensors based on low-power EPC-C1G2 RFID tags. The antenna and modulator/demodulator modules are common parts of any wireless communication device. The operation frequency of the circuit is limited by the clock control module and there is a module that efficiently and intelligently manages the power in the circuit. The chip is armed with several external sensors (e.g. temperature and humidity) and the multi-purpose sensor block includes internal sensors, controls all of them, and provides the interface with the rest of the circuit. Finally the EPC-C1G2 module supports lightweight cryptography primitives and the security protocol compliant with EPC-C1G2 standard. This last module is the one we have studied and implemented in this chapter. In [151] Zalbide et al. presented an ASIC implementation of the EPC-C1G2 standard with a sensor. We can compare the whole

Table 3.6: Hardware analysis of Chien-Huang EPC-C1G2 protocol.

Gate Equivalents (GE):

PRNG	32 bits	64 bits	128 bits
AKARI-1A	2840	5453	15197
AKARI-1B	2703	5109	9931
AKARI-2A	4185	8273	22089
AKARI-2B	3656	7036	13871
AKARI-2C	3685	6901	13516

Power (nW):

PRNG	32 bits	64 bits	128 bits
AKARI-1A	277	538	1347
AKARI-1B	280	528	1025
AKARI-2A	315	597	1442
AKARI-2B	323	616	1218
AKARI-2C	319	611	1192

Authentications/second

PRNG	
AKARI-1A	352
AKARI-1B	54
AKARI-2A	446
AKARI-2B	84
AKARI-2C	47

area of this passive sensing tag with the area used for our EPC module. For that comparison, the area of the cores are normalized following the approach described in [11], where T_A is the anchor of the transistor for the technology used, a A is the chip-area for T_a, and T_b is the technology to which the area is normalized.

$$A_{norm} = \frac{A}{(T_a/T_b)^2} \tag{3.1}$$

Using this approach and taking 1 nm as the reference technology ($T_b = 1\ nm$), the normalized area of the RFID sensor tag proposed in [151] is 19.42. In our case, for a 32-bit length compliant with EPC standard, in the worst case the EPC module occupies a normalized area of 1.57 and 1.66 for Burmester-Munilla and Chien-Huang EPC-C1G2 protocols, respectively. In other words, the EPC module uses less than 8.5 % in comparison to the whole chip area of the above mentioned RFID sensor

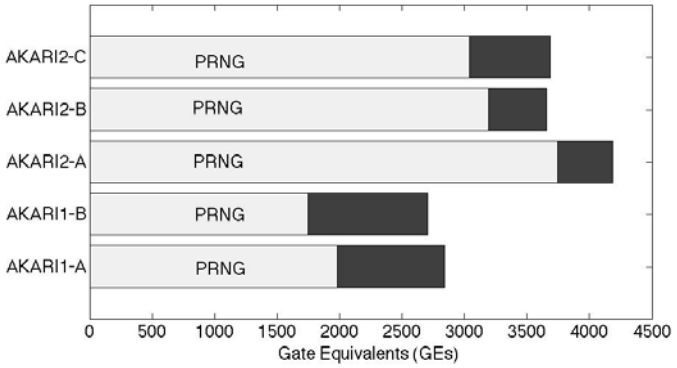


Figure 3.12: Area analyis of Chien-Huang protocol (32-bit architecture).

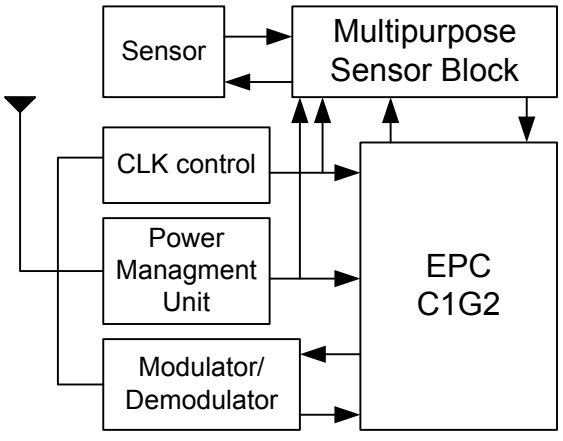


Figure 3.13: Block diagram of a passive sensing tag.

tag. This percentage could be even smaller since in our proposed EPC module the needed memory is counted in the module area calculations and a sensor tag is often armed with a external EEPROM and the EPC module could use this memory.

3.5 Conclusions

The necessity of implementing PRNGs has become more important since the EPC-C1G2 standard contemplates the implementation of these blocks. In addition to the conditions imposed by the standard, the requirements of the technology are very demanding. The pressing concern of miniaturization in order to implement massively RFID tags, makes that only 4000 GE can be devoted to security. Moreover, a high rate of read attempts is desired. Finally, as these kind of tags are passive, they must consume only a few nano-watts. All in all, design security devoted to RFID tags is a challenging task.

In this chapter, we have presented two new lightweight PRNGs suitable for constrained devices such as low-cost RFID tags. Several implementation architectures oriented to optimize some critical parameter have been proposed. In addition, two authentication protocols based on PRNGs have been studied in order to include our PRNGs in a complete security system. These authentication protocols comply with EPC-C1G2 standard. Finally, the synthesis metrics for each PRNG architecture and protocol are depicted.

Our experimentation suggests, as expected, that there are clear trade-offs between the circuit area and its throughput, so that optimization of one of them comes at the expense of a low performance in the other. We also found out that the PRNG area is a very significant fraction of the entire protocol implementation, which reinforces the view that advances in this area will greatly benefit from lightweight cryptographic components.

4

True Random Number Generators

True Random Number Generators (TRNGs) are one of the basic cryptographic blocks used in security protocols. These generators are commonly used to generate keys, and for that reason, they are one of the most important elements in a cryptographic protocol. Other applications of TRNGs include generation of nonces, padding plaintext or even countermeasures against side channel attacks (TRNGs are used to generate random noise within a certain bandwidth to avoid information leakage from some processes carried out by the system that can be exploited by an attacker).

Regarding RFID technology, TRNGs are mainly used in the key generation process. The random number is used as a PRNG seed in order to provide freshness in RFID systems [33]. TRNGs guarantee the tag's integrity because even if they are clonned, their output cannot be predicted. Regarding this feature, other typical application of TRNGs is defeating physical attacks at the same time that the non-traceability of RFID systems is enhanced [83].

Due to the importance of random numbers in security systems, the TRNG output should not only show good features like unpredictability, uniform distribution, etc. but also, reliability against environmental changes (with a malicious purpose or not).

Owing to the necessity of adding security in embedded systems, the TRNG integration in hardware devices has become an important challenge. These logic devices are generally designed to implement deterministic functions, therefore the inclusion of these kind of generators can have unwanted consequences. For that reason, it is important to carefully implement these kind of generators.

The TRNG implementation costs can be very high due to the variety of parameters (frequencies, place and route, power supply, post-processing necessities, etc.) that are

involved in the correct behaviour of the TRNG. These parameters are often adjusted in a trial and error process, for that reason, designing a TRNG is a challenging task.

Using Field Programmable Gate Arrays (FPGA) to implement TRNGs have become very popular nowadays. A singular challenge supposes the implementation of TRNGs on FPGAs as a consequence of the constrained resources of these sort of devices. FPGAs contain defined logic blocks and therefore, there is not flexibility when implementing designs. Furthermore, the vast majority of FPGAs do not include analogue blocks, which are often used in the generation of random numbers.

This chapter is structured as follows. First, an introduction that includes some generalities about the TRNG design process and evaluation methods is shown. In the state of the art subsection, several TRNG proposals and attacks are studied. In addition, a novel TRNG that will be later analysed and the coherent sampling technique which is used in our TRNG proposal are introduced. After that, in section 4.2,the randomness of a novel TRNG presented by Cherkaoui et al. is analysed, against different fault attack scenarios. In section 4.3, a new lightweight TRNG design based on coherent sampling is proposed.

4.1 Introduction

4.1.1 Design and evaluation

True Random number generators commonly use some kind of physical phenomenon as a source of entropy. Typically, these phenomena are analogue, so it is required to add some extraction mechanism in order to convert the analogue phenomenon in digital values that could be used by the device. Once the entropy source has been digitized, the statistical properties of the digitized signal will be evaluated with the purpose of establishing the TRNG quality. After the first evaluation, it is often necessary to add a post-processing block to correct the output distribution. Finally, due to the importance of the TRNGs in security systems, it is recommendable to check the quality of the random output. Some embedded tests are used to set an alarm when the generated number fails the tests. The typical blocks used in embedded TRNGs are depicted in Figure 4.1.

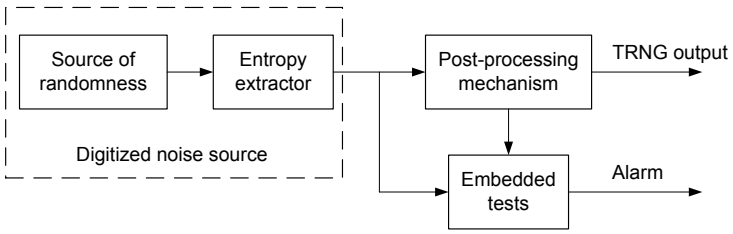


Figure 4.1: General scheme of a TRNG

Digitized noise source: As mentioned previously, FPGAs usually do not include an analogue block, consequently it is not possible to take advantage of analogue phenomena to extract entropy. In addition, these logic devices are designed to minimize any random behaviour,and as a result, it is not an easy task to implement TRNGs on FPGAs. The most common entropy sources used on FPGAs are metastability, delay variation between gates and thermal noise generated in the device.

- Metastability is typically defined as the phenomenon that is produced when the setup and hold time of a flip-flop are violated. The flip-flop output shows temporarily an undetermined value. After this phase, a final value is set randomly.
- The delay variation between gates is usually exploited by measuring the propagation delay of a signal. Typically, the deviation of the ideal behaviour (jitter) is used to extract entropy.
- Finally, as resistors and capacitors can be easily implemented in logic devices, the thermal noise generated in these components can be used to affect the frequency of a RC oscillator. The FPGA problem is that it does not include the possibility of implementing resistors or capacitors. Therefore this is not an entropy source suitable for FPGAs.

It is noteworthy that some FPGAs include phase locked loops (PLL). This analogue block can be used in the generation of random numbers [55]. As this block is not included in the majority of FPGAs, we will focus on other more general methods that can be implemented even on FPGAs without PLLs.

Post-processing: The vast majority of TRNG outputs present a slight bias before post-processing. This bias is caused by weaknesses in the entropy source or

a poor quality in the entropy extraction method. The bias presented before the post-processing makes the statistical test used to evaluate the TRNGs fail. In order to solve this problem, it is common to add a post-processing stage to obtain an acceptable bias at the final output. The most well-known post-processing techniques are the following:

- XOR corrector: The XOR corrector consists in hashing n consecutive input bits through a XOR operator to obtain 1 bit at the output. It is a very popular post-processing technique due to its straightforward hardware implementation and the fact that this corrector keeps the throughput constant. The main problem of this corrector is that if the input bits are correlated, the bias is not corrected at the output. A further analysis about the XOR corrector used as post-processing can be found in [41].

- Von Neumann corrector: This post-processing [141] is the most well-known technique because it generates unbiased outputs. Von Neumann corrector consists in processing pairs. If the two input bits are equal, they are discarded. On the other hand, the corrector output will be the first input bit. The main drawback is that a high percentage of the generated bits are discarded (typically about a 75%).

- Hash functions: Some authors include hash functions among the post processing techniques. According to [62], if the input data have a high entropy, the output generated by the hash function will be almost uniform. Besides, hash functions used as post-processing provide some interesting cryptographic characteristics as collision-resistance, one wayness, etc. The resources used to implement these functions are their main drawback. In addition, it is not easy to compare this technique with other post-processing methods.

- Good Linear Codes: this method is one of the most interesting options presented in the literature because this technique presents a good trade-off between resources and throughput. Furthermore, in [127] it is claimed that if an attacker knows n input bits, then he can not guess the output better. This feature provides more robustness to the system against bias introduced by an attacker. A complete study of these linear codes and their characteristics is presented in [80].

Tests: As TRNGs play a key role in cryptographic systems, they should be evaluated to guarantee at the output a stream of 0s and 1s uniformly distributed. The TRNG quality is directly related to the entropy source, and taking into account the post-processing, it is possible to select an entropy per bit rate at the output.

It is noticeable that some tests, that are used to evaluate TRNGs, have been designed to evaluate PRNGs. These tests only check the output distribution without regard to the phenomenon used to extract the entropy.

NIST, ENT and DIEHARD battery tests, that have been introduced in chapter 3, are widely used to evaluate the statistical properties of the outputs. Other well-known tests are the followings:

- FIPS: Federal Information Processing Standard is a standard used by USA government and developed by the National Institute of Standards and Technology (NIST)[53]. This standard specifies the security requirements of a cryptographic module used to protect sensitive information but unclassified. This test stablish 4 different security levels.

- AIS31: It is an evaluation methodology debeloped by Bundesamt fr Sicherheit in der Informationstechnik (BSI) [120]. This methodology was specifically oriented to evaluate TRNGs. Unlike the previous suites, AIS31 establishes a group of test to check the output before the post-processing block. This measure guarantees the quality of the entropy source. Furthermore, AIS31 determines the necessity of on-line tests that generate an alarm when the minimum entropy per bit will not be reached. Two different levels of security are defined, P1 and P2. Basically, P1 level is focused on the TRNG final output (after post-processing). P1 consist of 6 different tests (T0-T5) (a disjointness test(T0),an autocorrelation test (T5) and FIPS-140-1 tests (T1-T4). The security level P2 studies the raw output of the entropy source. It consists of 3 tests which are: A uniform distribution test (T6), A comparative test for multinomial distributions (T7) and an entropy test (T8).

- On-line TEST: In terms of security, the idea of implementing embedded tests that evaluate the quality of the TRNG (mandatory in AIS31) is very interesting. If there are enough resources, it could be a good option to implement some of the aforementioned tests. In typical application scenarios, the resources are

very limited. For that reason it is not possible to use these tests. In [119], a guide to choose efficient on-line tests for TRNGs was proposed. In lightweight cryptography, where each equivalent gate counts, it is important to select a test that observes a feature easily measured. Usually, tests that need a big continuous bit-stream are avoided. The Frequency test is often selected among the on-line tests due to its straightforward implementation and lightweightness.

4.1.2 State of the art

In this section some of the most recognized TRNGs presented in the literature are summarized. In addition, typical attacks against TRNGs are also depicted.

4.1.2.1 TRNGs on ASICs

As aforementioned, embedded devices as smartcards that are used in applications where high security is necessary, rely on TRNGs to provide security. TRNGs implemented on ASICs have the advantage of using tailored blocks as PLL or some other analog blocks. On the other hand, the TRNG design processes on ASICs are expensive, due to the variety of parameters that have to be tuned in order to obtain the required quality at the TRNG output.

Among the proposals reported in the literature oriented to low-cost RFID tags stand out [8], [29]. In [8], an oscillator-based TRNG is presented. The circuit relies on a system of jittered clocks that are being monitored by a clock arbiter-synchroniser, in order to have a random output signal. In [29] the TRNG consists of an analog random seed generator which uses the oscillator sampling mechanism and a LFSR for post-processing.

4.1.2.2 TRNGs on FPGAs

In this subsection TRNGs on FPGAs are presented. They have been grouped regarding their entropy source or the extraction method.

TRNGs using SRAM memories: In this group the TRNGs that use some features of SRAM memories to generate random numbers are contained. There are two main approaches: The first one uses the SRAM start-up state to extract

entropy ([65],[138]). The noise generated on the start-up process of the SRAM is non-deterministic. Using this noise as a seed in algorithms, it is possible to generate a stream of random bits. The main problem of this extraction method is that some SRAMs are reset on the start-up phase and it is not possible to take advantage of this phenomenon. The other method is known as *write collisions*([58],[60]). This method consists in generating a conflict in a particular address trying to write opposite values at the same time. This procedure generates a kind of metastable response that generates at the end a random bit. The problem of this extraction technique is that not all the addresses have a random response. For that reason an enrolment process is necessary. Moreover, the addresses can change their behaviour after a SRAM reset.

TRNGs using Metastability: Metastability in electronics is the ability of a digital electronic system to persist for an unbounded time in an unstable equilibrium or metastable state [25]. This phenomenon has been widely used to generate random numbers in ASICs. Taking advantage of metastability in FPGAs is more difficult due to the fact that these devices are designed to minimize metastability events. Vasyltsov et al. presented in [139] a TRNG that uses inverters (used as Ring Oscillators (ROs)) and multiplexers that switch between a metastable mode (entropy accumulation) and a oscillation mode. More recently, in [144], it has been proposed another TRNG that takes advantage of dual metastability. As mentioned before, the problem of using metastability is that FPGAs are designed to minimize it. Moreover, some vulnerabilities have been founded in [118].

TRNGs using jitter: As aforementioned, the jitter is the deviation of a signal ideal behaviour. Using jittery signals has become the most effective method to extract entropy in a FPGA. The jitter is composed of two main noises. Thermal noise and flicker noise. The thermal noise is the responsible for the non-deterministic component of the jitter. On the other hand, the flicker noise contributes with a deterministic component that depends on the manufacturing technology, the power supply, etc. There are two main techniques using jitter as a source of randomness:

- **Sampling jittery clock signals** This method consists in sampling high frequency signals shifted in phase using a D flip-flop. Sunar et al. reported in 2007 a new TRNG using this technique [127]. To the best of our knowledge, this was the first time that a stochastic model of a TRNG was presented. The

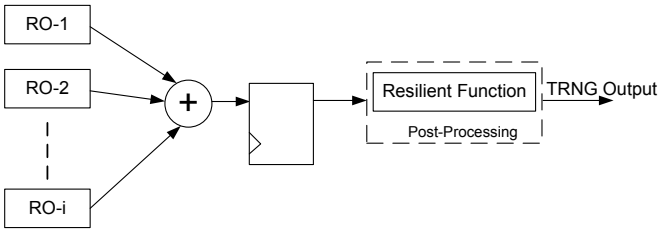


Figure 4.2: Sunar et al. TRNG

principle used by Sunar et al. is depicted in Fig 4.2. In this TRNG several ROs that are sampled at the same time are used. If at least one of these ROs is sampled in the jitter zone, a random bit will be generated. Using the provided stochastic model and a resilient function used as a post-processing block, a designer could select the entropy per bit. The main problem of this TRNG is that the XOR-tree and the D flip-flop are not able to handle all the transitions at their inputs.

In order to solve this problem, K. Wold et al. proposed in [147] a modified version that includes a D flip-flop after each RO. These flip-flops make possible to solve the problem of dealing with the transitions at the inputs. In addition, the authors claim that it is possible to remove the post-processing block and use less ROs. In [18], a study demonstrates that this TRNG presents pseudo-randomness at the output. In this study a simulation of the K.Wold design, with 18 ideal ROs (jitter-free), was carried out and it showed that a sequence that pass statistical tests was generated.

Two fault attacks have been reported in the literature against these proposals [90][14]. These attacks exploit the vulnerabilities of ROs.

Cherkaoui et al. presented a novel TRNG using this technique and Self-timed Rings [32]. Further analysis of this TRNG has been carried out in following sections due to the novelty of their proposal and its importance in this thesis.

- **Coherent Sampling** Coherent sampling is a technique that uses several clocks with related frequencies or phases. The technique is simple, a clock signal samples other clock signal in the edges. As these clock signals are not jitter-free, the output of the sampling process will be a random stream. Fischer and Drutarovsky presented in [54] a TRNG based on coherent sampling. In this design, two PLLs embedded in the FPGA were used to generate the clock

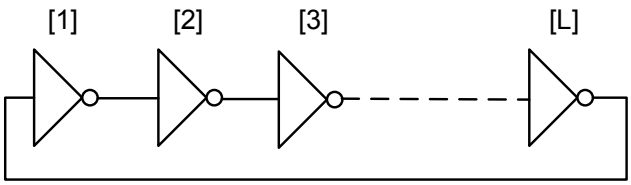


Figure 4.3: RO general architecture

signals. The weak point of this TRNG is that PLLs are not supported in all FPGA families. In order to solve this problem, Kohlbrenner et al. replaced the PLLs by two ROs that can be used in all FPGAs. Besides, they proposed a different sampler block. The problem of this TRNG is that an enrolment process is necessary to obtain the required frequencies at the RO outputs. In the same work, they claim that due to process variations, the normalized frequencies of ROs can differ in up to 8% inside the chip and even more from chip to chip.

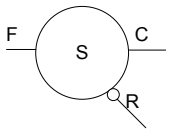
As coherent sampling is the technique used in our TRNG proposal, it is described more widely in section 4.1.2.5.

It is noteworthy that in almost all the proposals using these techniques, oscillators are used as entropy sources. Due to the importance of these oscillators in TRNG designs, in the next subsection are introduced the most extended oscillators used on FPGAs.

4.1.2.2.1 Oscillators used as entropy sources

Jittery clocks are usually used as entropy sources in many TRNGs. Especially in FPGAs where sources of randomness are very limited. Ring Oscillators (ROs) and Self-Timed Rings (STRs) are the most used alternatives.

- **Ring Oscillators:** ROs are the most widely used solution as generators of jittery clocks in both ASICs and FPGAs due to their low area, good integration in digital and analog design flow and important phase noise. Basically, a RO consists of a chain of an odd number of inverters or an inverter and delay elements connected to form a ring. The structure of a generic RO is depicted in 4.3.



F	R	C
0	0	C^{-1}
0	1	0
1	0	1
1	1	C^{-1}

Figure 4.4: Structure and truth table of a Self-timed Ring stage.

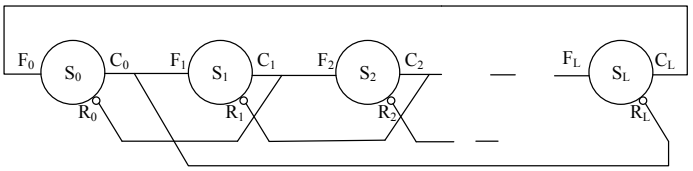


Figure 4.5: Self-Timed Ring structure.

ROs have been widely used in the literature [127] [147]. Their behaviour is well-known under some environmental variations (temperature, underpowering, etc) [149]. Among their weaknesses stand out the intra and extra device variability of frequencies. Several attacks that take advantage of the RO weaknesses have been reported in the literature [90][14].

- **Self-Timed Rings:**

  Self-Timed Rings (STRs) are well-known structures to generate clock signals in digital devices. A STR implements a handshake protocol that assures an evenly distribution of events through the different stages. For this, the implementation of a multi-phase oscillator based on a STR is highly configurable—frequency and phase resolution between signals can be chosen.

  - **Architecture**

    The basic element in a STR is a stage. Each stage consists of a Muller gate and an inverter, and implements the truth table shown in Fig. 4.4. If the forward input F, is different from the reverse output R, the output C takes the same value as F; otherwise the previous output is maintained.

    The STR architecture implements a micropipeline (ripple FIFO) introduced by Sutherland in [130] (see Fig. 4.5). The handshake protocol used guarantees the phase distribution between the micropipeline stages.

  - **Behavior and Configuration**

TTTTBBBB (01010000) → TTTBTBBB (01011000) →
TTBTBTBB (01011000) → TBTBTBTB (01100110) →
BTBTBTBT (11001100) → TBTBTBTB (10011001) →
BTBTBTBT (00110011) → TBTBTBTB (01100110) ···

Figure 4.6: Example of tokens and bubbles propagation in a Self-timed Ring.

In order to understand the STR operation we need to define the following parameters:

* L: Is the number of stages that compose the STR. Each stage can be initialized either to 0 or 1.
* $Tokens$ and $Bubbles$: A stage contains a token if its output C_i is not equal to the output $C_i + 1$. Converserly, a stage contains a bubble if its output C_i is equal to the output $C_i + 1$. The number of tokens (NT) and bubbles (NB) can be chosen during the initialization phase.
* N: It corresponds to the number of events distributed throughout the ring, which equals the number of propagating tokens in the ring.

A *token* will propagate to the next stage (s_{i+1}) if this stage contains a bubble. The *bubble* will occupy the backward stage s_i. The STR will have an oscillatory behavior if there are at least 3 stages, 1 *bubble*, and an even number of *tokens*. For example, in a 8 stage STR and a initial distribution of 4 *tokens* and 4 *bubbles*, the events will propagate as shown in Fig. 4.6.

It is important to note that the propagation delay of the ring depends on two analog phenomena: Charlie effects and drafting effects. We refer the reader to [146], where a complete model that explains both phenomena and how they affect the ring propagation delay can be found.

Regarding configuration possibilities, the frequency can be fine tuned in the initialization phase by changing the ratio between *tokens* and *bubbles*, that is, NT/NB. The maximum frequency is reached when:

$$\frac{NT}{NB} \simeq \frac{D_{ff}}{D_{rr}} \tag{4.1}$$

where D_{ff} and D_{rr} are the static forward and static reverse propagation delays, respectively.

The phase shift between two stages separated by n stages can be calculated as:

$$\varphi_n = n \times \frac{N}{L} \times 90^o \tag{4.2}$$

Note that if L is a multiple of N, some outputs will have the same phase, as it happens in the example shown in Fig. 4.6. In particular, there will be four different phases through the ring, with each phase appearing in two different stages ($stage_i$ and $stage_{i+4}$). Therefore, in applications of this oscillator in which the goal is typically to generate the maximum number of different phases, L mod N should not be equal to 0.

A comparison between ROs and STRs used as a source of entropy can be found in [30].

Others TRNGs: Owing to the importance of TRNGs in security systems, several TRNGs have been presented in the last decade. Several of them are not included in the previous classification but they are worthy to note. Dichtl et al. presented in [43] a TRNG that tries to transform pseudo-randomness in true-randomness. As LFSR is used as post-processing, it cannot be possible to assess a entropy per bit at the final output. Lozac'h et al. proposed a TRNG based on open loop chains and metastability in [84]. Finally, E.Bohl reported an interesting TRNG with on-line testability in [19].

4.1.2.3 Attacks on TRNGs

TRNGs have not only to generate a uniform distributed output without bias but also should be robust against attacks. There is an increasing interest in this topic due to the important role of TRNGs in cryptographic systems.

Firsts studies dealing with these problems in FPGAs where focused on the effects of temperature and voltage on the randomness. To the best of our knowledge, the first attack was presented in 2003 by M. Dichtl [42]. In this work [133],an attack against a TRNG that uses free-running oscillators and two LFSR was presented. An attacker takes advantage of controlling the environmental conditions to guess the frequency of the oscillators. Once the frequency is known, it is easier to guess the LFSR internal state.

In 2008, Sunar et al. presented a detailed report [149] about the effects of temperature and voltage in their TRNG [127]. It was concluded that ROs frequencies are dependant on the power supply and temperature, while the randomness was barely affected.

R. Santoro et al. presented [118] in 2009 an evaluation of temperature effects and high electronic activities around three TRNGs [127], [139] and [43]. It can be concluded that [139] and [43] present weaknesses when the temperature is increased or there is a high electronic activity around the TRNG.

It is worth to highlight the work presented by A.Markettos et al. [90]. In this study the authors performed a fault injection attack against the K.Wold's TRNG. The ROs were locked to the same phase by coupling the injection frequency on to the power supply of the device. If all the ROs have the same phase, the TRNG will generate several deterministic bits. This attack was carried out using ROs implemented in discrete logic.

Following the work started by A.Markettos et al., in [14],an electromagnetic (EM) attack against Sunar's TRNG was carried out, where the EM effects in the phase of ROs to control the TRNG output were exploited.

As shown in [90] and [14], ROs are a weak point in TRNGs and their use should be avoided.

In conclusion, a good TRNG should be portable among different FPGAs families, should be robust against attacks and should be lightweight (area, power and clock cycles).

4.1.2.4 Introduction to Cherkaoui et al. TRNG.

A. Cherkaoui et al. design is introduced in this section due to the novelty of this proposal. To the best of our knowledge, vulnerabilities of this TRNG have not been found yet. For this reason, in this thesis we have explored the response of a lightweight implementation of this TRNG in typical TRNG evaluation scenarios.

A. Cherkaoui et al. presented in [32] a novel TRNG design that exploits the jitter on a Self-Timed Ring to generate random numbers.

The STR consists of L-stages that uses a handshake request and acknowledgement protocol in order to guarantee the propagation of events (N) simultaneously. In

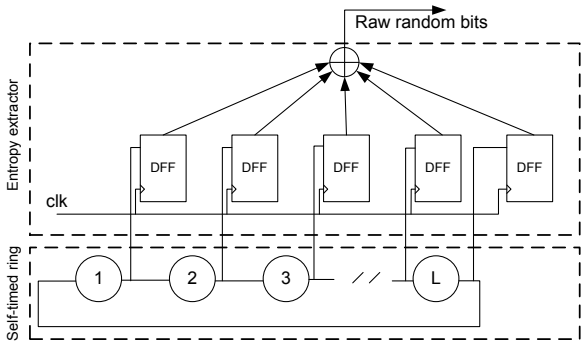


Figure 4.7: Core architecture of an RNG based on a Self-Timed Ring (STR) [32]

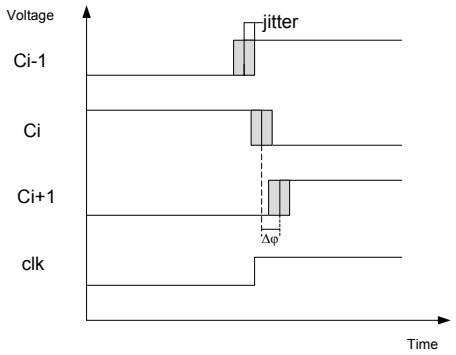


Figure 4.8: Entropy extraction principle

the STR configuration process, L and N are selected to be co-prime. This STR configuration guarantees the generation of L equidistant phases and a phase resolution $\Delta\varphi = T/2L$, where T is the period that can be tuned precisely choosing the ratio N/L. In Fig. 4.7, the architecture of the TRNG is presented.

Fig. 4.8 illustrates the entropy extraction principle. The STR output signals are re-indexed according to their mean arrival time (C_i and C_{i-1} are not adjacent stages). Since each signal C_i is sampled using the same reference clock clk, if the jittery interval around the mean signal phase is longer than the phase difference between two signals C_i and C_{i-1}, at least one signal is sampled in its jittery time interval. The resulting sample then has a random value, and hence the output of the XOR gate is also random. A complete stochastic model to provide a lower bound of entropy per bit as a function of the ring characteristics (number of stages, oscillation period, and jitter size) is reported in [32].

As post-processing, an n^{th}-order parity filter was used. This filter combines n

successive input bits into one output bit using a XOR function, which enhances the entropy per output bit, but reduces the throughput by n. The main advantage of the parity filter is that combined with the proposed stochastic model, it enables simple entropy per bit correction. The variable n_{pmin} that represents the minimum filter order necessary to pass the statistical tests was defined.

Regarding the TRNG hardware, each STR stage contains a Muller gate and an inverter implemented in one look-up-table (LUT). Hard-wired connections between the LUTs and adjacent flip-flops were used to connect each stage with its corresponding flip-flop. Ring stages were placed so that the delays between adjacent stages were identical, or at least similar (ring topology). Interconnecting each stage with the previous and the following one, a ripple FIFO is created. Finally, the FIFO is closed to create a ring that implements the handshake protocol [130]. A XOR-tree was selected to hash the sampled outputs in only one bit. This XOR-tree was implemented using a ripple structure (registers are used between each XOR row) in order to achieve high working frequencies.

The evaluation of this TRNG was carried out in two different FPGAs (Altera Cyclone III and Xilinx Virtex 5). Several architectures (different number of stages) were evaluated. AIS31 statistical suite was used to test the TRNG output. Main parameters,($\Delta\varphi$, T, n_{pmin},etc) for the different architectures and both devices can be founded in [32].

4.1.2.5 Introduction to Coherent sampling.

Due to the importance of coherent sampling on this thesis (Coherent sampling technique has been used in our TRNG proposal.), the principles of coherent sampling are introduced in this section. After that, we present the main TRNG proposals that exploit this technique.

4.1.2.5.1 Background

Coherent sampling is a well-known technique to sample periodic signals at finer time intervals. Coherent sampling refers to an integer number of cycles that fits into a

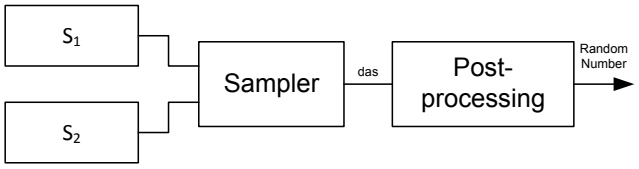


Figure 4.9: General architecture of a TRNG based on coherent sampling.

predefined sampling window. Mathematically, this can be expressed as:

$$\frac{f_{in}}{f_{sample}} = \frac{N_{cyc}}{N_{samples}} \tag{4.3}$$

where f_{in} is the sampled signal (S_1) frequency; f_{sample} is the sampling signal (S_2) frequency; N_{cyc} is the number of cycles of the sampled signal; and $N_{samples}$ is the number of samples.

If N_{cyc} and $N_{samples}$ are high and co-primes, the repetition period of samples will be maximum—that is, we will have the highest resolution of the sampled signal. This is an interesting feature because if the number of periods (frequencies) is constant for ideal sources of S_1 and S_2, in physical systems where these clock signals contain jitter, this number will be random because of the Gaussian random component contained in the jitter.

The general architecture of a TRNG using coherent sampling is depicted in Fig. 4.9. The signal S_1 would be sampled by the signal S_2, generating a digitized analog signal (known as "das"). If the quality of the raw output is not high enough, a post-processing stage is added to guarantee a uniform output. A mathematical model of physical RNGs based on coherent sampling can be found in [16].

4.1.2.5.2 TRNGs based on Coherent Sampling

The first time, to the best of our knowledge, that coherent sampling was used in an FPGA to generate random numbers was in [54]. In that work, Fischer et al. used a PLL embedded in an Altera FPGA in order to guarantee the relation between N_{cyc} and $N_{samples}$. As explained in Section 4.1.2.2, the main drawback of this proposal is that the TRNG is not portable to other FPGA vendors. Besides, PLLs are not supported in all FPGAs.

In [78], Kohlbrenner and Gaj replaced PLLs by ROs with the aim of obtaining a portable design for FPGAs from different vendors. The RO frequencies are selected to be close but not identical. The RO outputs are connected to a sampler circuit that generates a stream of 0's and 1's. The length of this stream is counted module 2 to generate a random bit. The weakest point of this design is that it requires a very complicated manual placement and routing process in order to finely set the ring frequencies. This is a consequence of the high variation (up to 7 %) among the RO frequencies in the same FPGA. To overcome such a sensitivity to placement, the authors suggest a design with four ROs that are sampled by a fifth one.

In [37], Cret et al. take up the basic idea of using only two ROs. In this design, the authors introduce a multiplexer to alternate the sampling signal. They claim that the placement sensitivity is overcome using a parametrizable post-processing. The main weakness of this TRNG is that the quality of the raw output, without the post-processing stage, is really poor. In addition, Cret et al. present the cycle lengths of the signal generated in the sampler and its distribution is not an evidence of the claimed randomness—which is actually far away from an uniform distribution.

Finally, in [137] the authors present three designs based on different clock generators for different FPGA models. More precisely, the generators are RO-RO, RO-PLL (for Altera FPGAs), and RO-DFS (for Xilinx FPGAs). Apart from the technology dependency, the pair RO-RO cannot be fully automated since its design needs manual placement and routing. Finally, it is worth mentioning that the authors introduce the interesting idea of generating one random bit per half period by using mutual sampling.

4.2 Analysis of a Novel TRNG

As aforesaid in the previous section, the TRNGs reported in the literature are not suitable enough to secure lightweight applications due to different factors.

Some of them are not portable among different FPGA families because of the lack of a special block, e.g PLLs on the Fischer and Drutarovsky design [54]. Others like the TRNG based on metastability presented by Vasyltsov et al. in [139] has been attacked successfully [118].

Sunar's design ([127]) was a very promising alternative due to the fact that was the first TRNG that provided a stochastic model. But the attacks performed against the ROs in [90] and [14] affect directly to the validity of the stochastic model.

In August, 2013, A. Cherkaoui et al. presented a new TRNG design keeping the main idea of Sunar's design of sampling signals in the jitter zone [32]. ROs have been replaced by a Self-Timed Ring in order to avoid their vulnerabilities. The authors present a realistic stochastic model that guarantees a rate of entropy per bit. As far as we know, this TRNG does not have any known vulnerability (problems related to RO phase interlock and violations of the stochastic model) and no-attacks have been reported in the literature.

In this section, in order to study if this design presents any vulnerability, we have explored the response of the TRNGs in several typical scenarios. Underpowering, temperature-variation susceptibility and power glitches have been considered. In addition, for the first time (to the best of our knowledge), a clock glitch attack against the TRNG has been performed. Specifically, against a "lightweight" implementation of this TRNG affecting XOR-tree weaknesses. The implementation of this lightweight version is important because area can be an exclusion factor to select this TRNG.

This section is structured as follows. In Section 4.2.1, we describe the threat model and related work on that topic. In Section 4.2.2, the experimental setup is introduced. Section 4.2.3 presents the obtained results for different fault-attack injections and discussed the results. Conclusions are drawn in Section 4.2.4.

4.2.1 Threat Model

Two kind of attacks can be considered on TRNGs. First, *passive* attacks that collect data from the TRNG output (before and after the post-processing) in order to predict future values. Second, *active* attacks that try to modify the TRNG behavior to control its output. Active attacks can target the randomness source, the entropy extractor, algorithm post-processing, or in the alarm generated by embedded tests. In this section, we present an attack that targets the entropy extractor, specifically through the violation of timing constraints.

Nowadays, high frequencies are demanded in almost all applications. Hence, the reduction of the critical path delay is one of the most important issues in the IC

design. This critical delay is the minimum time necessary to process the data through the combinational logic between two D-registers sharing the same clock. The clock period has to be higher than this critical delay in order to guarantee the correct circuit behavior. This critical delay depends on the delay of the combinational gates and also depends on the set-up and hold time of the registers.

The violation of time constraints induce faults in the circuit. There are two main methods to achieve timing violations:

- Overclocking: This technique consists in decreasing the clock period to achieve a time violation. If the clock frequency is high enough, it can be observed how faulty data is latched by D flip-flops. To make certain the correct behavior, the clock period T_{clk} must be:

$$T_{clk} > T_{critical} + T_{set-up} \tag{4.4}$$

  where $T_{critical}$ represents the delay of the critical path and T_{set-up} represents the setup time of the registers. From an attackers point of view, it is important to control exactly when the fault is injected. Clock glitches are well known in practice to induce timing violations. Typically, an attacker needs to be able to control two parameters: When the fault is injected and also the duration of this fault (consecutive clock cycles that are affected by the glitch injection). Recent works have induced faults to cryptographic algorithms [153, 6, 82] using this technique.

- Increasing propagation time: The other way to induce faults through timing violations is to increase the critical path delay ($T_{critical}$). In [112], the equation of the inverter propagation time t_{pLH} was presented, using first order analysis of the dynamic behavior:

$$t_{pLH} = \frac{C_L \left[\frac{2|V_{th,p}|}{V_{DD}-|V_{th,p}|} + ln\left(3 - 4\frac{|V_{th,p}|}{V_{DD}}\right)\right]}{\mu_p C_{ox} \frac{W_p}{L_p}(V_{DD} - |V_{th,p}|)} \tag{4.5}$$

  where C_L is the load capacitance, $V_{th,p}$ represents the PMOS threshold voltage, V_{DD} is the power supply voltage, μ_p is the mobility, C_{ox} is the gate oxyde capacitance, and (W_p/L_p) denotes the aspect ratio of the PMOS. For more

complex combinational logic, the aforementioned equation becomes more complicated, but the inversely-proportional relationship between propagation delay and power supply voltage still holds true. That means, the propagation time through any combinational logic will increase with a decrease of V_{DD}. The same effect is caused by increasing the temperature, as temperature is directly related with mobility μ_p.

Underpowering is generally used to induce fault by timing violation due to the fact that any decrease of V_{supply} will suppose an increase of the propagation delay [112].

Once again, an attacker needs to control precisely when an attack is performed. For that reason, underpowering is not an efficient way to achieve malicious goals. Power glitches consisting in a sudden negative change of voltage have been used to induce faults in cryptographic algorithms [153, 12]. With this procedure, a transient fault is caused by increasing the propagation delay in combinational logic. In summary, underpowering induces a permanent increase of the propagation time, and a power glitch induces a transient increase of the propagation time.

Increasing the temperature in order to induce faults is also an attack scenario usually considered.

As it is shown in Fig. 4.10, in the TRNG presented by Cherkaoui *et al.* the critical path is in the XOR-tree. We note that as the STR oscillator is purely combinational, clock glitches do not affect its behavior. For this specific TRNG, clock glitches can be used also to violate the conditions presented in the stochastic model, sampling the same jitter realization twice by generating a glitch which meets the following statement:

$$F_{clk} > 1/2\Delta\varphi. \tag{4.6}$$

In practice, this violation is very difficult to achieve because the clock glitch frequency induced should be of a few GHz (22 GHz for a configuration of 63 stages). Probably the XOR-tree critical path delay will have a bigger period. For that reason, a fault would be introduced before a violation of the stochastic model appears.

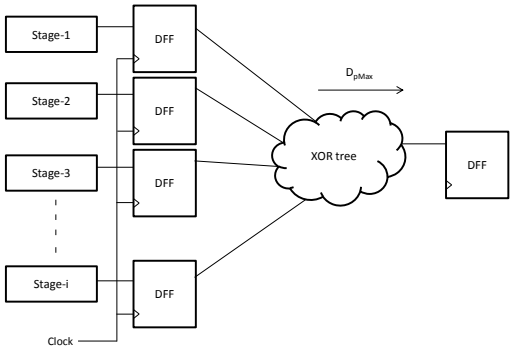


Figure 4.10: The representation of the TRNG by Cherkaoui *et al.* with the path delay of the XOR-tree (D_{pMax})

4.2.1.1 Related Work

Typical scenarios where the TRNGs are evaluated include the randomness evaluation for different temperatures and core voltages. Surprisingly, to the best of our knowledge, only two physical attacks on random number generators have been presented in the literature. The first attack was presented by Markettos and Moore [90]. This attack aims at an RO based TRNG implemented in an IC. Injecting a sine wave onto the power supply, the operating conditions were modified and a bias appeared at the output signal. The other attack, presented in [14], targets another RO based TRNG [147] using an electromagnetic attack. In this attack, the ROs were locked on the injection frequency, generating a controllable bias at the output.

On the other hand, power and clock glitches have been widely used to attack cryptographic algorithms. Clock glitches are a well known technique to induce faults in cryptographic designs. Several platforms have been designed to induce this kind of faults [6, 46]. To the best of our knowledge, this kind of fault injection has not been considered as a serious threat for TRNGs. As it is shown in the following sections, clock glitches should be considered in the typical evaluation scenarios. The fault injection using power glitches is widely used in microcontrollers, but only a few papers reported the use of power glitches in FPGAs [153]. Besides, as in the clock glitches case, this kind of attack has not been considered on TRNGs until now.

4.2.2 Implementation and Experimental Setup

We have implemented several STR configurations in a *Spartan-6 XC6SLX45* and a *Spartan-3 XC3S1000*, that are both from XILINX. Following the recommendation of [31], a hard-macro for each FPGA has been designed to guarantee the phase distribution at the register inputs. The hard-macro implements an inverter, a Muller gate using a 6-input LUT and also the sample D flip-flop in the same slice. In addition, we have tried to avoid bottleneck effect using a placement process that maintains the delay of consecutive stages. The XOR-tree is implemented using 6-input LUTs in order to save area.

The results presented in Table 4.1 were obtained using a core voltage of 1.2 V. The sampling frequency was set to 60 MHz. This frequency has been chosen because a bit rate of a few Megabit per second (>10 MHz) is typically sufficient for normal applications. More demanding applications like 10 G-bit Ethernet servers would need up to 20 Mbits/s that can be reached with a 60 MHz clock. Low Voltage Differential Outputs (LVDS) have been used to avoid circuitry effects. Specifically, as an opposite current flows in the two wires, the electromagnetic fields are canceled each other. Therefore, the generation of electromagnetic noise is reduced. The STR period was measured using an oscilloscope (the *LeCroy WavePro 725Zi*) and an active differential probe with a 1 GHz bandwidth. The sampling rate was set to 2 GS/s to obtain accurate results. Furthermore, a simple software post-processing filter that enhances the entropy per bit was implemented. This post-processing consists of a XOR filter that combines n consecutive bits into one output bit. This filter was implemented in Matlab, which was also used for analyzing the measured (and filtered) power traces.

We applied the statistical test of NIST [114] to 100 sequences of 10^6 bits for each STR configuration. NIST provides a test suite that consists of a statistical package including 15 tests which evaluate the randomness of binary sequences. For the test, we set the confidence level to 0.01. Raw data were acquired using a FIFO memory and a RS232 protocol at 19.2 Kb/s. The n_{pmin} value presented in Table 4.1 shows the filter order necessary to pass successfully the NIST test.

In terms of T, $\Delta\varphi$, and n_{pmin}, these results are very similar to the results presented in the original paper [32]. The difference in the throughput is due to the fact that we are using a clock of 60 MHz instead of a 400 MHz clock.

Table 4.1: Results for different STR configurations in Spartan FPGAs

Device	STR		Measurements		Compressed Data	
	L	N	T	$\Delta\varphi$	n_{pmin}	Throughput
Spartan-6	63	32	2.88 ns	22.8 ps	7	8.5 Mbit/s
	127	64	3.18 ns	12.5 ps	5	12.0 Mbit/s
	255	128	3.52 ns	6.9 ps	3	20.0 Mbit/s
	511	256	4.27 ns	4.1 ps	3	20.0 Mbit/s
Spartan-3	63	32	3.77 ns	29.9 ps	5	12.0 Mbit/s
	127	64	3.84 ns	15.1 ps	3	20.0 Mbit/s
	255	128	3.99 ns	7.8 ps	2	30.0 Mbit/s
	511	256	4.21 ns	4.1 ps	-	60.0 Mbit/s

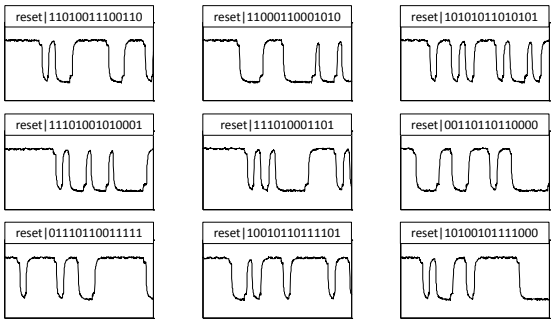


Figure 4.11: Nine output sequences captured after restarting the TRNG. Note that all sequences are different

In order to characterize the pseudo randomness properties of this TRNG, we decided to apply the idea of Dichtl *et al.* [43]. The idea is to restart the TRNG from the same initial conditions. In Fig. 4.11, nine oscillograms of repeated restarts are presented. The horizontal axis represents the time and shows the first 350 ns (corresponding to the generation of 14 bits using a clock of 40 MHz). The vertical axis is the voltage of the output signal. It is clearly visible that several random signals are generated after the same restarting point.

4.2.2.1 Setup for High-Temperature Fault Injections

We used a Positive Temperature Coefficient resistor heater (PTC) in order to increase the temperature from room temperature (about 35°C) up to 85°C, *i.e.*, the maximum operating temperature for the Spartan-3. A PT-100 sensor was used to measure the temperature. In order to facilitate the heat propagation, several layers of conductor

material were used between the heater and the FPGA, as it is shown in Fig. 4.12. A USB transfer protocol at 60 Mb/s was used to collect the data from the FPGA as fast as possible (to acquire the data at the same temperature).

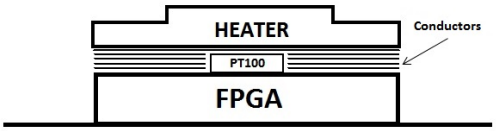


Figure 4.12: Setup for tampering with the temperature of the FPGA. A *PT100* is placed between heating element and FPGA for measuring the temperature.

In Fig. 4.13, the frequency response of the STRs against temperature changing is shown. A Spartan-3 with a core voltage of 1.2 V was used to carry out the measurements. It shows that the frequency of the STRs decreases with temperature increase. At 35°C, the frequency of the analyzed STRs is between 235 and 265 MHz depending on the number of stages (63, 127, 255, and 511) while at 85°C it is between 230 and 260 MHz.

We also analyzed the frequency response of a ring oscillator, which consists of three inverters, in order to compare it with the behavior of the STRs. As a result, we observed the same effect as obtained for the STR: the frequency decreases the higher the temperature.

4.2.2.2 Controller Board and FPGA Extension Board

In order to tamper with the supply voltage as well as with the clock signal, a custom-made fault-injection controller board was used, shown on the right side of Fig. 4.14.

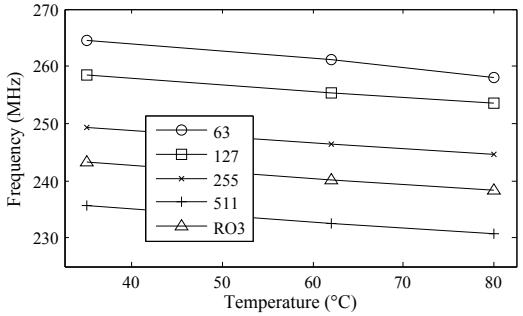


Figure 4.13: Frequency of the STR measured at different temperatures. For higher temperatures the frequency decreases.

The main part of the controller board is a XILINX Spartan-6 XC6SLX45 FPGA. For inserting clock glitches into the clock signal, a similar approach to the one presented in [6, 46] is applied. In order to insert power glitches, a high-speed multiplexer with different, adjustable voltages at the inputs is used. Power-supply voltages between 0 V and 5 V are supported. The output of this multiplexer is connected to the supply pin of the attacked device. Switching among the different inputs allows to tamper with the supply-voltage level. The parameters glitch duration, glitch shape as well as the point in time when the glitch is injected are configured by a PC that is connected via USB.

In order to allow attacking a wide range of devices, the controller board provides many pins for connecting extension boards. For the experiments conducted during this work, an FPGA Extension Board (FEB) was developed as shown on the left side in Fig. 4.14. Most relevant connections between the controller board and the FEB are: a power supply, clock signal, trigger signal, and a serial connection for communicating with the PC. With the trigger signal, a synchronization between the FEB and controller board can be performed, which allows a precise glitch injection according to the execution of the implementation on the FEB. The FPGA placed on the FEB is the same as on the controller board, *i.e.*, a XILINX Spartan-6 XC6SLX45.

As underpowering would be used to induce time violation faults, we have studied the behavior of the STRs for different power-supply voltages. We have varied the core voltage between 0.70 V to 1.26 V (note that according to the Xilinx Spartan-6 specification [148] underpowering happens if the core voltage is lower than 1.16 V). Also note that the minimum voltage for which the FPGA can be programmed is 0.70 V.

As Fig. 4.15 shows, the core voltage affects directly the frequency of the STRs. Hence, $\Delta\varphi$ is affected. The reduction of $\Delta\varphi$ does not affect the randomness, even a higher frequency could be used for sampling the STR outputs fulfilling the stochastic model constraints. Comparable results were presented in [31].

4.2.3 Experimental Results

In this section, experimental results are presented. First, we provide the results of the temperature susceptibility of the TRNG. Second, we provide results of performed

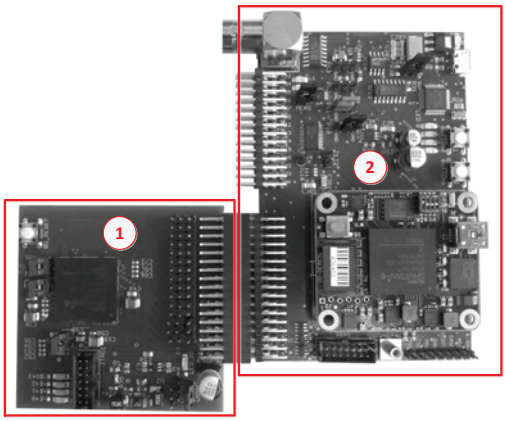


Figure 4.14: (1): FPGA Extension Board (FEB); (2): Controller Board.

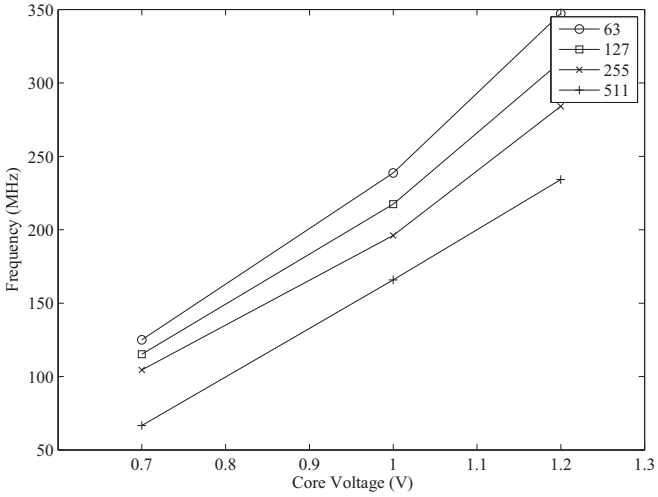


Figure 4.15: Frequency of the STR measured for different core voltage values. Reducing the core voltage decreases the frequency.

underpowering attacks. Finally, we present results of power and clock-glitch attacks.

4.2.3.1 High-Temperature Fault Attacks

In our experiments, we increased the temperature of the targeted Spartan-3 FPGA to 35, 45, 55, 65, 75 and 85 °C. We did not increase the temperature further to avoid destruction of the FPGA (so we operated the device within the specified maximum operating conditions). Note that we aimed to evaluate the impact of an increased ambient temperature on the output of the TRNG and not to cause over-heating faults through operating the device beyond the maximum ratings as, for example,

recently shown by [71, 24]. The Spartan-3 FPGA was powered with a core voltage of 1.20 V and was clocked with a frequency of 60 MHz. We increased the temperature from two different starting temperatures: from a lower and a higher temperature of the goal temperature in order to test the TRNG reliability against past conditions. We applied the NIST test to 100 sequences of 10^6 bits and obtained similar results in terms of n_{pmin} to those presented in Table 4.1. However, the increased temperature did not show any effects on the randomness of our TRNG. The increase of the XOR-tree propagation time is not enough to induce a fault in the circuit. In particular, the STR frequency decreases with the increase of the temperature as it is shown in Fig. 4.13. This frequency decrease means a phase resolution ($\Delta\varphi$) increase which makes less likely to sample at least one signal in the jitter zone. In this case, the increase of the $\Delta\varphi$ is negligible (which is an increase of 1 ps in the worst case). In addition, thermal noise, which is responsible for the random component of the jitter, increases with the increase of the temperature. For that reason, this phenomenon compensates the loss of randomness due to an increase of the phase resolution. So the combination of both phenomenon did not significantly affect the TRNG and randomness of the output.

4.2.3.2 Underpowering

Using the controller board and the FEB (Spartan-6 FPGA), we measured the frequency response for different STR configurations (63, 127, 255, and 511) and operated the device with a clock frequency of 20, 40, and 60 MHz. The power supply was set to either 0.70, 1.00, and 1.20 V. We acquired 100 sequences of 10^6 bits for each different combination of frequencies, STR configurations, and core voltages.

Using underpowering, the same phenomena are observed as in the temperature fault-attack scenario. The propagation time increase through the XOR-tree does not have any impact in the TRNG normal operation. On the other hand, the phase resolution increase is more important in this case because there is a significant difference among the frequencies reached for different core voltages (Fig. 4.15). Table 4.2 presents the $\Delta\varphi$ obtained using a core voltage of 1.00 V and 0.7 V. The largest difference in terms of phase resolution is obtained for the 63-stages architecture, reaching a $\Delta\varphi$ difference of up to 41 ps. This change in the phase resolution is non-negligible and provokes a bias that can be measured before the post-processing.

For the 63-stages architecture and a core voltage of 0.70 V (worst case), a 2.6 % bias (more 1's than 0's) appears before the post-processing. However, this bias is eliminated by the post-processing filter. In summary, using underpowering generates random bias bits that are corrected after the post-processing without a visible impact, in terms of randomness, at the final output.

4.2.3.3 Power-Glitch Attacks

Power glitches were injected for different frequencies, core voltages, and STR configurations. We successfully show that faults are induced in the TRNG output that are caused by timing violations. First, we present results for a nominal supply voltage. Afterwards, we present results of underpowering attacks. It is important to notice that for both power glitch scenarios, the STR was still oscillating without any changes at the output.

- **Nominal Supply Voltage** We set the power supply voltage to 1.20 V and injected power glitches to different STR configurations. As it was expected, using low clock frequencies (4, 8, 20, and 32 MHz), the power glitches do not impact the final output due to the fact that the critical path delay period is still lower than the clock frequency. However, using a 40 MHz clock and a power glitch length of 10 000 clock cycles, *i.e.*, 250 μs, a bias appears at the TRNG output before the post-processing is done. This little bias is filtered by the post-processing block and it is not noticeable at the final output. Fig. 4.16 shows the precise instant were the power glitch is induced creating a slight bias.

Table 4.2: Period and Phase resolution for the different STR configurations using a core voltage of 1.00 V and 0.70 V

Voltage	L	T	$\Delta\varphi$
1.00 V	63	4.20 ns	33.2 ps
	127	4.60 ns	18.1 ps
	255	5.10 ns	10.0 ps
	511	6.03 ns	5.8 ps
0.70 V	63	8.00 ns	63.4 ps
	127	8.68 ns	34.1 ps
	255	9.57 ns	18.7 ps
	511	15.00 ns	14.4 ps

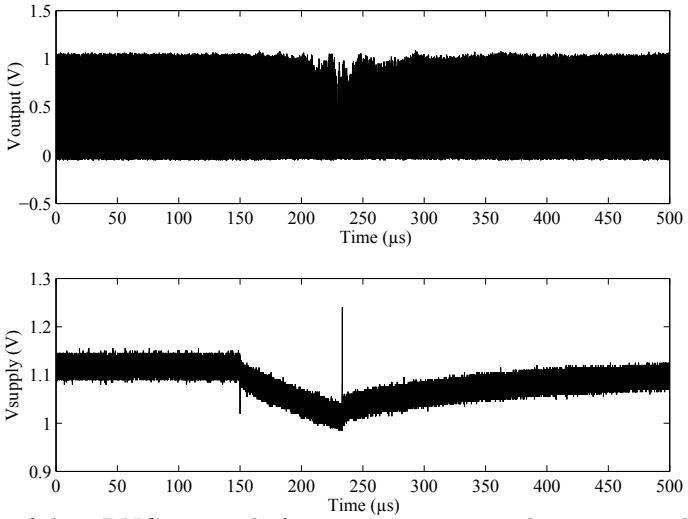


Figure 4.16: Bias of the TRNG output before post-processing when a power glitch with a length of 87.5 μs and a core voltage of 1.20 V are used. No bias is observable after post-processing.

A configuration of 255-stages, 40 MHz frequency, and a power-glitch length of 3 500 clock cycles, *i.e.*, 87.5 μs, were used to generate the figure. The maximum glitch length that can be induced without erasing the FPGA configuration is 750 μs.

- **Underpowering** We set the supply voltage to only 0.7 V and induced power glitches during the TRNG computation. This induces a permanent increase in the critical path delay period without modifying the clock period. Due to that reason, we observed that the TRNG implementation is more sensitive to power-glitch attacks. No evidence of faults were observed for clock frequencies between 4 MHz and 20 MHz. Using a clock of 32 MHz and a minimum glitch length of 62.5 μs, faults were induced for different TRNG configurations. We note that the maximum power glitch length that can be used before deprogramming the FPGA is 375 μs. Fig. 4.17 shows the TRNG bitstream, before post-processing, under different glitch lengths. A 255-stage TRNG was selected, using a 40 MHz clock. Fig. 4.17a shows the output in normal conditions, in Fig. 4.17b a power glitch of 62.5 μs was induced, and Fig. 4.17c presents the output affected by a 187.5 μs power-glitch length.

The effects of power glitches are observable after the TRNG post-processing phase. For a visualization, we used a clock frequency of 40 MHz and induced a power glitch with a length of 62.5 μs on different STR configurations. Fig. 4.18a shows the output

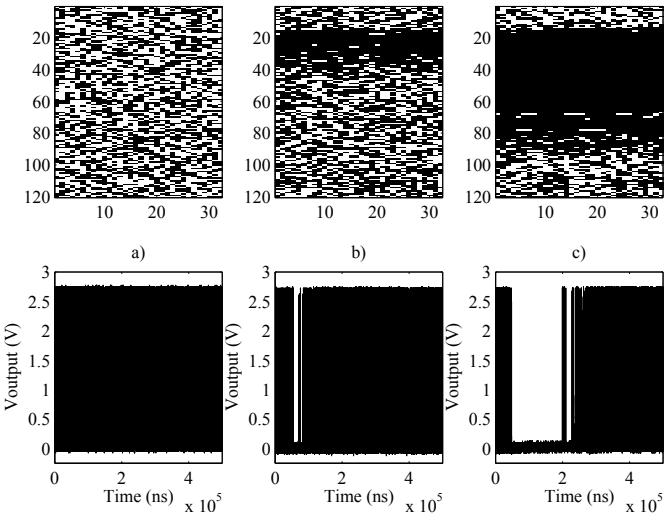


Figure 4.17: TRNG output before post-processing (lower plots) and resulting bit-stream (upper plot) for a core voltage of 0.70 V. For case (a) no power glitch was inserted, for case (b) a power glitch of length 62.5 μs was inserted, and for case (c) a power glitch of length 187.5 μs was inserted.

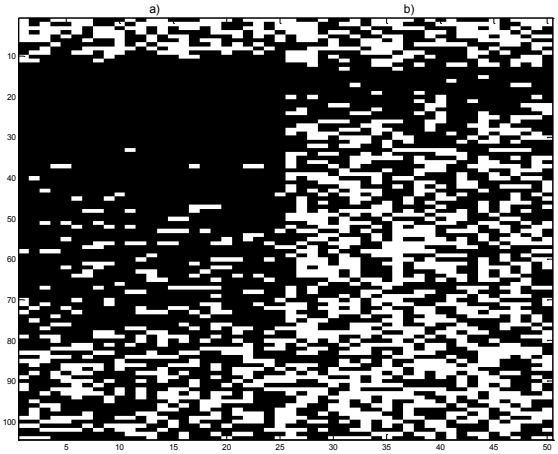


Figure 4.18: Final bit-stream for a core voltage of 0.70 V and a power glitch length of 67.5 μs. For case (a), 3-XOR filter with a configuration of 255 stages was applied, for case (b), 7-XOR filter with a configuration of 63 stages was applied.

after a 3-XOR filter for a configuration of 255 stages. Fig. 4.18b shows the TRNG output after the post-processing of a 7-XOR filter for a configuration of 63 stages. As the 255-stages configuration offers a higher entropy rate per bit, it is necessary to apply a more "lightweight" post-processing step than for the 63-stages configuration.

As the same number of bits before the post-processing is affected by the power glitch, after the post-processing the number of bits affected will be higher in the

architecture that needs a less complex post-processing (255-stages) than in the architecture that uses a more complex post-processing (63-stages). This observation has a significant impact on a frequently used on-line test, the longest-run test. Low implementation costs make this test a popular choice. Longest-run tests count the number of equal consecutive bits at the TRNG output and if this number exceeds a predefined threshold, an error signal is generated in order to signalize the faulty behavior. As a consequence, a TRNG implementation using the less-complex post-processing (255-stages) might not pass this test in presence of a power glitch while an implementation applying the more-complex post-processing (63-stages) might pass this test in presence of the similar power glitch. For example, if 100 bits are affected by the power glitch before the post-processing, the output will have 14 consecutive equal bits for the 63-stages architecture and 33 consecutive equal bits for the 255-stages architecture. If the threshold of the longest-run test is *e.g.* set to 20, the first case will be detected as normal behavior (the number of consecutive equal bits is smaller than the threshold value) while in the second case an error signal will be generated (the number of consecutive equal bits exceeds the threshold value). An attacker could take advantage of this situation controlling a bunch of bits undetectable for the long run test. In this case, using more post-processing steps can suppose to be more vulnerable. For that reason, using a XOR filtering is not a good option.

4.2.3.4 Clock-Glitch Attacks

The controller board allows us to induce clock glitches accurately and generate a one-bit faulty output. For different frequencies (4, 8, 10, and 20 MHz) and a core voltage of 1.20 V, clock glitches were injected without observable effect in the output. This is due to the fact that the highest frequency reached during the clock glitch for 30 MHz is about 60 MHz. At this frequency, the statement $T_{clk} > T_{critical} + T_{set-up}$ is met.

For a STR configuration of 255 stages, setting the core voltage to 1.20 V, and using a clock frequency of 40 MHz, clock glitches were injected generating a faulty output in the TRNG. In Fig. 4.19, two different numbers of cycles are affected by clock glitches (10 and 55, respectively). The operation frequency used is 40 MHz, obtaining up to a frequency of 90 MHz during the clock glitches. We also have tried

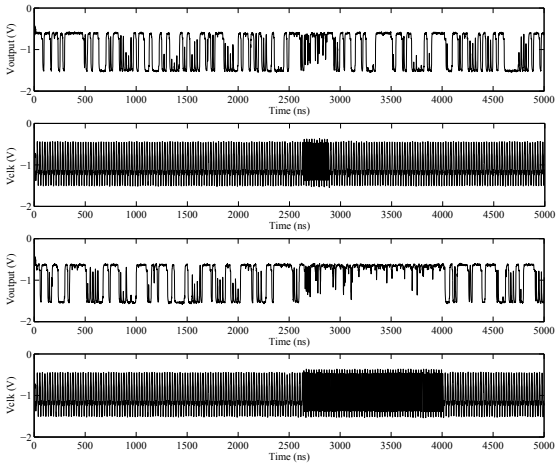


Figure 4.19: TRNG output for different number of cycles affected by a clock glitch (upper plots: 10 cycles, lower plots: 55 cycles).

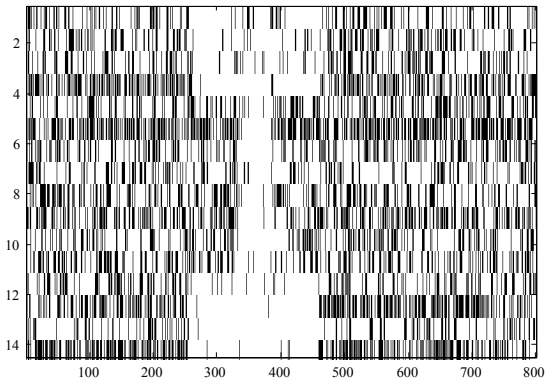


Figure 4.20: Bit-stream of the TRNG precisely modified by using clock glitches with different lengths.

to use underpowering (0.70 V) and obtained similar results.

Fig. 4.20 presents the result of several successful clock-glitch injections using a clock of 40 MHz, a core voltage of 1.20 V, and a STR configuration of 255 stages. With this plot, we demonstrate that we were able to fully control the output of the TRNG by changing the number of clock cycles affected by the clock glitches. We also controlled the interval between the trigger event and the time instance when clock glitches are injected. As a result, we drawn an "I" letter into the bit stream.

In summary, using clock glitches it is possible to control precisely the output of the TRNG, changing the numbers of bits desired when it is necessary. With this kind of attack, we can introduce faults in the instant of a key generation, or bypass easily

some on-line tests that are executed from time to time. Clock glitches induced faults are a serious threat to TRNGs and should be included in typical attack scenarios.

4.2.3.5 How to Thwart These Attacks?

The XOR-tree has proved to be the weak point of the analyzed TRNG. In order to avoid the effects of power and clock glitches, it is necessary to enhance the implementation of this XOR-tree by using a ripple structure. For our experiments, we therefore added a register after each XOR row that decreases the critical path delay. This costs some additional resources (area and power), for example, it is necessary to add 51 registers for the STR configuration with 255 stages. These 51 registers are added after each 6-input LUT that implements a 6-XOR operation. In Fig. 4.21, a representation of the XOR-tree ripple structure is represented.

A. Cherkaoui *et al.* used this ripple structure in order to reach higher frequencies up to 400 MHz (they did not use it to provide protections against fault attacks). However, this XOR-tree ripple structure can also be applied to increase the security level of TRNG implementations by lowering the critical path delay and sensitivity window for power and glitch attacks. Note that this does not prevent high-frequency

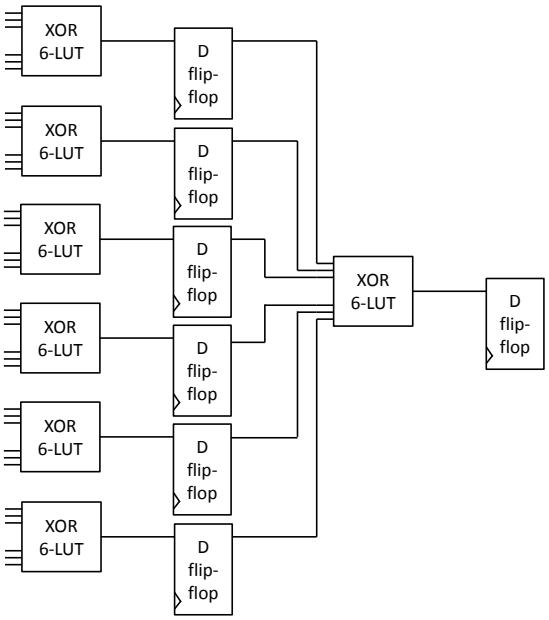


Figure 4.21: Implemented XOR-tree ripple structure.

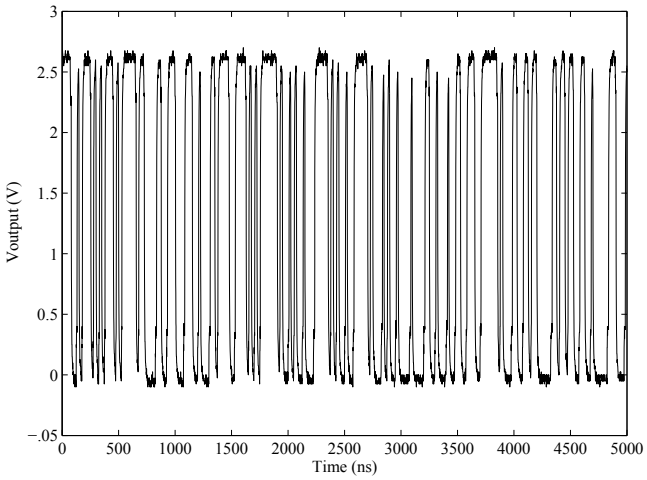


Figure 4.22: TRNG output of the enhanced design under normal operation conditions.

glitch attacks but protects against glitches at least below 90 MHz as shown in this section and demonstrated in our experiments.

In order to demonstrate the reliability of our protected implementation, the same attacks have been carried out. Fig. 4.22 shows the normal behavior of a 255-stages STR using a core voltage of 1.20 V and a clock frequency of 40 MHz. It shows that the used ripple structure does not affect the normal operation of the TRNG. The figure shows the first 200 bits that were generated by the TRNG.

In Fig. 4.23, the result is shown where a power glitch was induced using a core voltage of 1.20 V and a glitch length of 87.5 μs.

Fig. 4.24 shows the result of a similar setup where a power glitch was induced using a core voltage of 0.70 V and a glitch length of 62.5 μs. Both figures show that the enhanced TRNG architecture does not provide any bias before the post-processing step. Thus, the random behavior is not affected by the performed attacks (and attacking parameters).

Fig. 4.25 shows the TRNG output of two different attacking setups. The upper plot shows the output in a setup where a clock glitch has been induced during a clock period of 55 cycles. The lower plot shows the result of a clock glitch that has been induced during a clock period of 10 cycles. In contrast to the scenario presented in Fig. 4.18, there is no bias at the TRNG output. A clock frequency of up to 90 MHz has been achieved. The same result was obtained using underpowering the device using a core voltage of 0.7 V only.

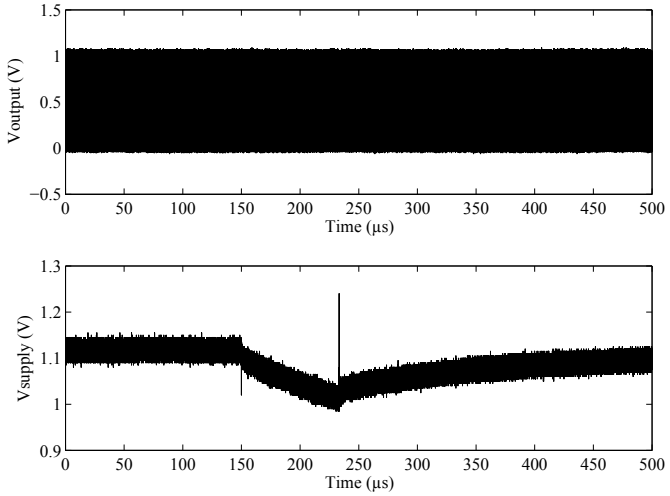


Figure 4.23: TRNG output before post-processing when a power glitch with a length of 87.5 μs and a core voltage of 1.20 V are used. In contrast to the standard design (cf. Fig 4.16) no bias is observable.

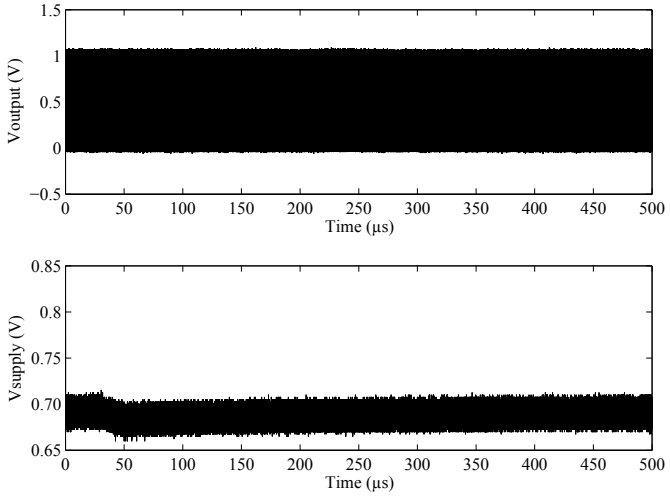


Figure 4.24: TRNG output before post-processing when a power glitch with a length of 62.5 μs and a core voltage of 0.70 V is induced. No bias is observable.

Fault attacks based on clock glitches are only possible if the attacked device receives the clock signal from an external clock source. That means, if the clock signal is generated on-chip, the previous attacks based on clock-tampering are not applicable.

If the device has an external clock pin, it is possible to include an clock-observation circuit as presented in Fig. 4.26 in order to ensure a glitch-free clock signal (clk_{gf}) for the D flip-flops of the XOR tree. For that purpose, one D flip-flop clocked with

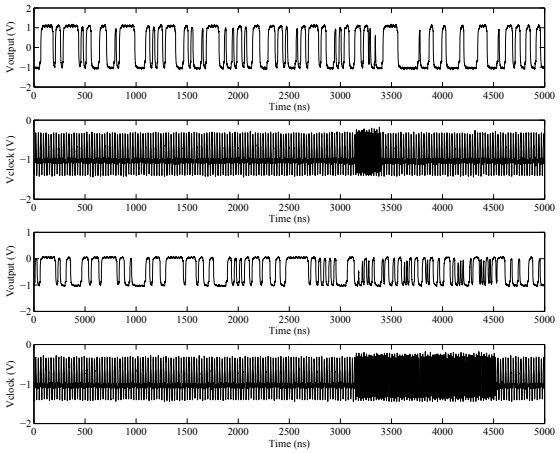


Figure 4.25: TRNG output of the enhanced design for different number of cycles affected by a clock glitch (upper plots: 10 cycles, lower plots: 55 cycles). No effect on the TRNG output is observable.

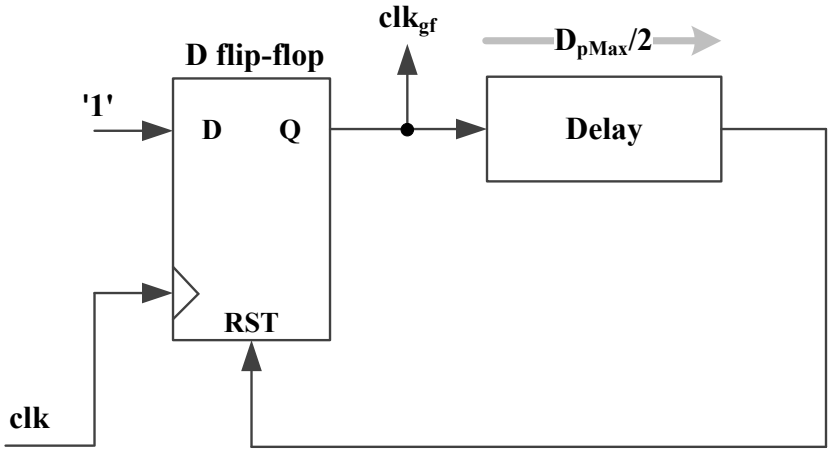


Figure 4.26: Generation of a glitch-free clock signal for the D flip-flops in the XOR-tree.

the standard clock signal clk and the input D (that is permanently set to one) is used. The output Q is connected to the reset of the flip-flop via a delay element. The output of this element changes $D_{pMax}/2$ after an input change. The output Q also serves as clk_{gf}. This circuit prevents clock glitches shorter D_{pMax} to propagate to the D flip-flops of the XOR-tree.

This glitch-free clock generator has been synthesized in a Spartan 3E. To implement the delay element, we used an inverter chain. This solution offers a straightforward implementation and also portability among different FPGA families. Moreover, the response of an inverter chain is well-known in different scenarios (underpow-

ering, temperature variations, etc). Other approximations have been taken into account, like PLLs, but they have been discarded because of their complexity and non-portability.

In comparison with the ripple structure, our proposed glitch-free clock generator is even more lightweight. For the STR configuration with 255 stages, 50 registers are saved in comparison to the ripple structure implementation. In addition, the correct operation of the XOR-tree is guaranteed under any clock glitch attack (which is not the case for the ripple structure).

4.2.3.6 A General Formalization Model

We present a general formal model to guide designers and evaluators to systematically test their architectures against this kind of attack. For this purpose, we need to specify the time frame when the architecture is generally susceptible to "overclocking" and subsequently define all fault-injection parameters that could lead to a bias of the TRNG output. This allows us to propose a testing strategy for evaluators to guarantee security against this attack under the assumptions of this underlying model.

Definition 1 (Time Frame T_{frame})

We define the time frame when the device is susceptible to overclocking attacks as $T_{frame} = T_{clock-to-Q} + T_{critical} + T_{set-up} < T_{clk}$, where $T_{clock-to-Q}$ represents the delay from the clock transition to a possible output change of a flip-flop and T_{set-up} represents the time that a flip-flop input signal needs to be stable before the sampling is triggered by the next clock edge. T_{frame} is the time starting at the beginning of a positive clock edge until the time when the last signal propagated through the critical path of the design and all registers are stable and settled. Intended changing of the clock signal during T_{frame} causes overclocking and a possible bias.

Remark 1

In our model, we assume a synchronous CMOS logic. We also consider (active) single-edge-triggered clocking, which is not a loss of generality though and can be

easily extended to two-phase clocking circuits, for instance. For simplicity, we also do not consider effects such as clock jitter and skew that also impact T_{frame}.

Definition 2 (Fault-Injection Model)

An adversary is able to induce additional clock-signal edges or to change the signal-propagation delay of the internal circuit. The changing of the clock signal is arbitrary in the sense that he/she can define the time when the additional clock signal is injected and how the additional clock-signal shape looks like, *i.e.*, the starting time and duration of the signal transition.

This fault model implies a physical bound in the fault-injection frequency which depends on the used fault-injection setup. An adversary is therefore only able to inject faults with a maximum frequency $f_{max} = 1/\Delta$, where Δ denotes the minimum time between two consecutive glitch injections. Furthermore, we define $T_{min} = \lambda \cdot \Delta$ to be the minimum time between the rising clock edge and the rising clock edge of the injected glitch and λ is a multiple of Δ. Besides this, we do not make any restrictions regarding higher-order fault-injections. An adversary is also able to induce several clock glitches during T_{frame} which causes several overclock signals and thus causing faults in two or more consecutive instructions.

Changing the propagation delay can be caused by intentionally varying the ambient temperature or by varying the power-supply conditions (*e.g.*, underpowering).

Definition 3 (Clock-Glitch Injection Time)

For clock glitches, we denote $t_{start} \in [T_{min}, T_{frame}]$ the time when a positive clock-glitch is injected.

Definition 4 (Signal-Propagation Delay)

We define room temperature as the minimum temperature denoted by $Temp_{min} \cong$ 23°C. Since an increase of the signal-propagation delay is only caused by heating, we define the maximum ambient temperature to be the maximum temperature where the device is barely able to produce the correct output, *i.e.*, $Temp_{max}$. Regarding power-supply modifications, we denote $0 \leq V_{min} \leq V_{nominal}$ as the minimum voltage

level where the device is barely able to produce the correct output (can be also set to zero in the tests), and $V_{min} < V_{nominal} < V_{max}$ the maximum voltage level (typically beyond the recommended ratings but lower in order to avoid destruction).

4.2.3.7 Testing Strategy

We propose to apply fault attacks that are performed in the given parameter ranges described in Definition 3 and 4. All fault attacks are applied during the sensitivity window T_{frame} (given in Definition 1). If all tests pass (under these attacks), the design is considered secure against attacks up to the specified adversary frequency f_{max} under the assumptions given in this model (with $\lambda = 1$).

4.2.3.8 Discussion about the Obtained Results

We have shown that our TRNG implementation (an implementation of the recently published TRNG architecture of A. Cherkaoui *et al.*) is resistant against temperature variations in the range of 35 °C to 85 °C. Note that the temperature evaluation is one of the typical tests carried out for TRNGs. In [118], the response of three TRNGs were measured, showing an influence in terms of randomness for 2 of the three TRNGs evaluated.

Another typical testing case to evaluate TRNGs is by applying different core voltages. In [149], a complete study of the influence of variations in the power supply on ROs was carried out. These variations affect the randomness of the TRNG depending on the number of inverters the ROs are composed of. For our analyzed TRNG implementation, we have shown that it is resistant against this kind of attack.

Moreover, we have shown that power glitches can induce faults in the TRNG but we were not able to fully control which bits should be biased and which bit should be not. However, we have successfully injected faults in our TRNG implementation by underpowering the FPGA device. These results emphasize the need of countermeasures against this kind of attack.

Finally, we have shown that clock glitches can produce very precise biases in the TRNG. We were able to control and influence each bit of the TRNG output thus making this kind of attack one of the most powerful attacks against our analyzed

TRNG implementation. Power and clock glitches pose a serious threat for TRNGs and should be considered in typical evaluation scenarios.

In order to thwart these attacks, we integrated a ripple XOR-tree structure into the TRNG and evaluated the performance. It showed that this structure helps against clock-glitch fault attacks below clock-glitch frequencies of theoretically 400 MHz (note that we evaluated glitches up to 90 MHz only). This might be a good solution to prevent many clock-glitch based attacks that are performed with standard fault-injection equipment. More sophisticated fault-injection attacks with fault frequencies beyond 400 MHz could still induce faults and should be considered in future work.

In addition to this proposal, a glitch-free clock generator that prevents clock glitches shorter D_{pMax} to propagate to the D flip-flops of the XOR-tree, has been presented. This clock generator is a lightweight and portable solution that guarantees the correct operation under clock glitch attacks.

As an outcome of these investigations, we recommend to apply a more secure post-processing technique like BCH-codes [80] as for example applied in [127]. BCH-codes are a well-known alternative with low-area requirements and an acceptable penalty in the throughput. As in particular presented in [127], a BCH-code was used in the TRNG which provided good results. The code had a compression factor of 16 and could be implemented using 256 registers and a few XORs. The main advantage of using BCH-codes, acording to [80], is that this kind of post-processing is more reliable than XOR filters or Von Neumann correctors against an adversary bias.

To the best of our knowledge, this is the first time that power and clock glitches are injected in a TRNG implementation. It is interesting to point out, as shown before, the critical path of our design lies in the XOR-tree, so other TRNGs that use very similar architectures, *e.g.*, [127, 147], are expected to be vulnerable against these kind of fault attacks as well.

4.2.4 Conclusions of the analysis

In this section, we performed different fault attacks on a TRNG modified implementation of A. Cherkaoui et al. [32]. This modified implementation is suitable for low cost environments. The implementation was running on two FPGA platforms (a Spartan-3 and a Spartan-6). We induced various power and clock glitches dur-

ing TRNG operation by varying the glitch location and duration. We successfully demonstrated that an adversary can to cause a bias in the random output of the TRNG. This bias can be exploited in attacks to reveal the exact value of the random number and thus extract secret key information of implementations of cryptographic algorithms. We also highlight the simplicity of these attacks and that they can be performed with low cost. We therefore suggest to include the applied fault attack tests in standard evaluation scenarios to evaluate the resistance against these types of attacks.

Furthermore, we evaluated our TRNG implementation against thermo attacks and underpowering. We varied the ambient temperature of the FPGA and evaluated the impact of temperature on the random outputs. We also powered the FPGAs below the recommended power-supply specifications to evaluate the resistance against faults. For both attacking scenarios, we did not observe any bias in the output. The implementation resists against these types of attacks and guarantees the stochastic model validity.

Finally, we tested the original version of the TRNG that introduces several registers after each XOR row in order to obtain a higher performance. These registers reduce the critical path delay. This ripple structure can be used as a simple countermeasure against most power and glitch attacks that are performed below a certain frequency (which is also often limited in practical attacks). We performed glitch attacks below 90 MHz under various operating conditions and showed that the enhanced TRNG implementation provides resistance against these attacks.

In conclusion, Cherkaoui et al. original proposal is secure against the fault attacks presented in this section. On the other hand, the lightweight modified implementation presents some vulnerabilities. For that reason, we can conclude that Cherkaoui et al. proposal can not be implemented in resource-constrained systems.

4.3 A New TRNG based on Coherent Sampling with Self-timed Rings

As aforesaid, the desired features for a TRNG suitable for resource-constrained environments are lightweightness, robustness and portability. Motivated by these

goals, authors have reported in the literature several TRNGs designs.

In previous sections, several TRNGs that are not suitable for resource-constrained devices for different reasons have been presented.

For instance, regarding the portability, a TRNG based on Coherent Sampling technique (CS) was presented in [54]. It is a lightweight design but can not be implemented in some FPGA families (no portable) because it uses a Phase-Locked Loop (PLL).

Concerning robustness, Sunar et al. presented a promising TRNG suitable for FPGA [127].Nevertheless, this design was rapidly discarded since it suffers from implementation problems, mostly related to the number of signals handled by the XOR-tree. Moreover, the quality of the raw signal is rather poor and needs post-processing. In addition, in [14] and [90] ROs vulnerabilities to frequency injection were exploited, in which the ROs are locked to an injected frequency and the jitter phenomena as source of randomness is neutralized.

The Kohlbrenner and Gaj proposal ([78]) can be considered insecure because uses ROs, and additionally it is not portable, as the design depends so much on the selected device that it has to be tuned (manual placement and routing) for each FPGA implementation.

Finally, introduced in section 4.1.2.4, the TRNG presented by Cherkaoui et al. [32] is secure and portable but uses a substantial amount of resources in terms of power and circuit area. This renders it unsuitable for constrained devices. More specifically, the STR generates 63 signals that are sampled and finally passed through an XOR-tree, generating a high activity, and, correspondingly, having a high power consumption. Moreover, the hardware requirements are superior to the ones that can be afforded in most constrained devices. In the previous section a lightweight implementation was tested but some vulnerabilities appeared.

In this section, we present a new TRNG based on the coherent sampling technique (4.1.2.5). On the one hand, the design takes advantage of some key STR features, which help us solve the implementation problems (mainly the device dependence) suffered by Kohlbrenner and Gaj's design [78], while simultaneously allowing us avoid the vulnerabilities linked to the use of ROs. On the other hand, the proposed design is very efficient in hardware, which makes it suitable for devices with limited capabilities, and offers a relatively high throughput. The remaining of this section

is organized as follows. In Section 4.3.1, our proposal is presented together with some necessary implementation considerations. The experimental results, both about the randomness quality and hardware requirements, are presented in Section 4.3.2, together with a comparison between our proposal and the most relevant designs. Finally, Section 4.3.3 concludes the section and summarizes our main contributions.

4.3.1 Our Design

The design presented in this section is inspired by the TRNG proposed by Kohlbrenner and Gaj in [78]. We use the same architecture but replace the ROs by 2 STRs. Cherkaoui et al. carried out in [30] an exhaustive comparison between ROs and STRs. According to this work, the main differences are:

- STR robustness to voltage variations can be enhanced by adding more stages. ROs do not offer this feature.
- STRs present a lower extra-device frequency variation when operating at high frequencies.
- In STRs the period jitter does not depend on the number of stages but it is mostly dependant of the jitter generated in each stage.

From the security point of view, these features are very interesting. In fact, in [30] the authors conclude that STRs are more robust to attacks than ROs, and this property is inherited by our proposal. Furthermore, replacing ROs by STRs provides our design with the possibility of having at least L different signals in each STR. Each one of those L signals can be used as a sampling or sampled signal, since each stage can be considered as an independent source of entropy—the number stages is equal to the number of independent entropy sources. Moreover, the STR is highly configurable, being easy to set desired frequencies for the STR outputs.

4.3.1.1 Architecture overview

Fig.4.27 and Fig.4.28 show the different blocks that make up our TRNG. Fig.4.27 depicts the two STR used in our design. Both STRs are composed of L stages that generate L different outputs with a frequency f_{STR}. The number of *tokens*

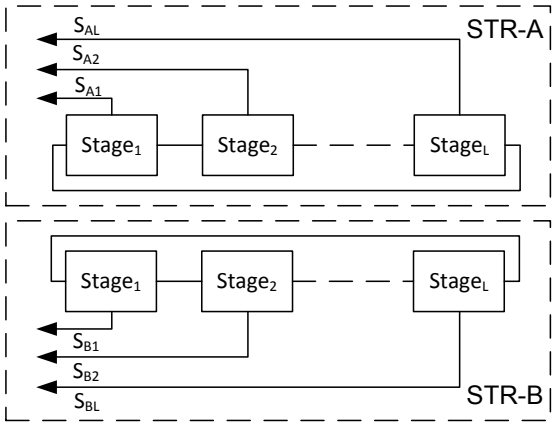


Figure 4.27: Self-Timed Ring structure of our TRNG.

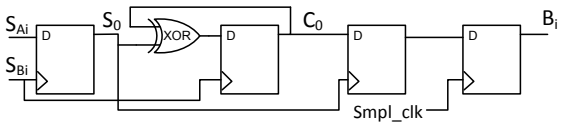


Figure 4.28: Sampler structure of our TRNG.

and *bubbles* are selected in the reset phase attending to the frequency and phase necessities.

The jitter contained in the STR outputs is extracted using the sampler circuit shown in Fig.4.28. Each sampler circuit is composed of 4 D-type flip-flops and 1 XOR gate. The first flip-flop uses the signal S_{Bi} to sample the signal S_{Ai}. The signal S_0 will be high while the rising edges of S_{Bi} occur during the high level of S_{Ai}. In Fig.4.29 we show the behavior of S_0 taking into account that S_{Bi} contains jitter. As consequence of such a jitter, the cycle length of S_0 will not be constant.

In our design, both signals, S_{Bi} and S_{Ai}, contain jitter. As a variation of the original sampler design that includes a one-bit counter latched by S_0, in our design we use the simplified version presented in [137]. In this scheme, instead of counting the cycles of S_{Bi}, we count the number of cycles that S_0 is at a high level. If such a number of cycles is even, the previous output is maintained; otherwise, the output changes. Two D flips-flops and 1 XOR gate are involved in this process. Finally, the last flip-flop samples the signal C_0 using an external clock. This external clock determines the TRNG throughput. As our design is composed of two STRs with L

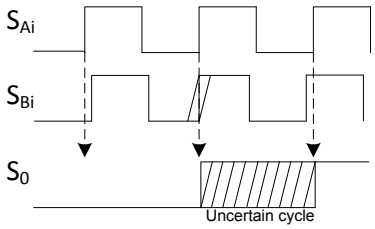


Figure 4.29: Sampler behavior.

stages each, L sampler circuits are necessary (see Fig. 4.9).

Finally, our design includes a post-processing unit that might be needed depending on the quality, in terms of randomness, of the raw data. The selected post-processing is a parity filter, which has been widely used as post-processing in previous proposals such as [32] and [37]. More precisely, a n^{th} parity filter takes n consecutive bits and XOR all them together to produce one bit. This post-processing offers a simple bias reduction with the penalty of a throughput reduction—the filter reduces the bit generation by a factor of n.

4.3.2 Experimental results

We have implemented the proposed TRNG in a FPGA Spartan-3E XC3S500E. Two 8-stage STRs have been implemented and configured in the reset phase to obtain a STR output frequency of 300 MHz. Several frequencies have been used in the external clock that samples the signal C_0. Eight bits are generated with each rising edge of the sampling clock.

In order to obtain almost the same propagation delay in the different stages, a hard-macro has been designed. This hard-macro implements, using a single Look-Up Table (LUT), a Muller gate and an inverter. If Altera's FPGAs are used instead, these hard-macros can be implemented using their equivalent hard-wired macros.

To show evidence of the Gaussian jitter in the STR outputs, we have counted the number of cycles of the signal S_0, as done in [137] and [78]. Fig.4.30 depicts the time evolution of the S_0 length (at the top of the figure) and a histogram of the cycles (at the bottom). The histogram population corresponds with 1 300,000 measurements. The average period of S_0 is 38.69 ns with an standard deviation of 0.215 ns. As the

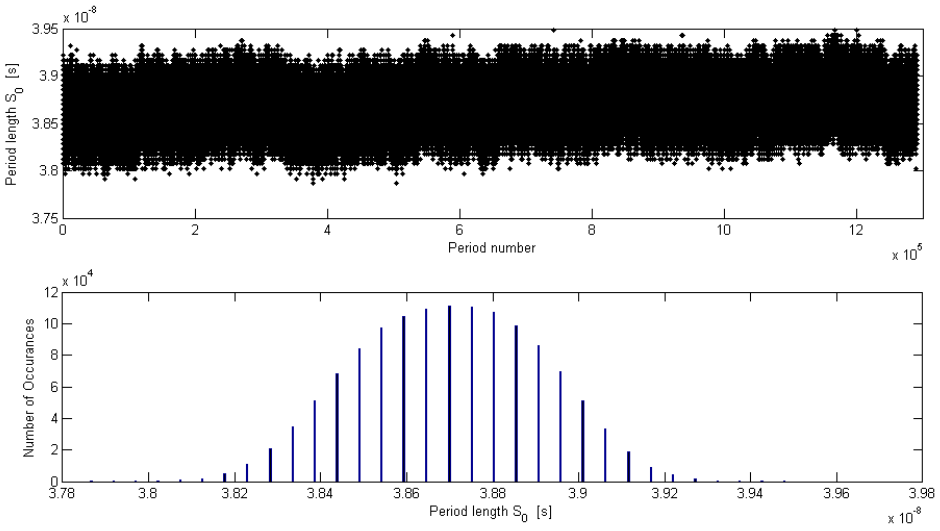


Figure 4.30: Time evolution and histogram of S_0.

Table 4.3: Experimental results: Pass Rate (PR) proportion and average p-value (PV) for generated traces.

	0.5 MHz		1 MHz		5 MHz		10 MHz		25 MHz		50 MHz	
	PR	PV	PR	PV	PR	PV	PR	PV	PR	PV	PR	PV
b1	97,91	0,41	98,58	0,44	98,41	0,57	99,08	0,55	99,5	0,58	93,83	0,22
b2	98,41	0,43	99,33	0,34	82,58	0,18	82,5	0,30	84,66	0,22	39,66	0,16
b3	98,83	0,46	99,08	0,62	98,58	0,59	98,91	0,41	98	0,44	99,33	0,60
b4	99,66	0,59	99,41	0,45	99,33	0,36	99,41	0,45	99	0,59	83,41	0,17
b5	92,66	0,25	87,16	0,20	98,5	0,59	98,25	0,55	99,16	0,39	97,91	0,30
b6	98,41	0,54	98,75	0,47	31,66	0,04	31,58	0,01	31,25	0,08	22,25	0,05
b7	98,75	0,36	99,16	0,45	98,83	0,53	98,91	0,61	98,58	0,36	66,58	0,24
b8	99,16	0,39	98,91	0,31	98	0,44	97,5	0,40	96,83	0,43	48,5	0,15
Total	97,97	0,43	97,55	0,41	88,23	0,41	88,27	0,41	88,37	0,39	68,93	0,24

frequency of the STR has been set to 300 MHz, that means that the average cycle length is 11.6090 cycles. In conclusion, the histrogram distribution clearly shows evidence of the underlying randomness in the sampling process, and, by extension, in each stage of the STR.

We have chosen two STRs with 8 stages since this configuration is easily tunable and offers a good trade-of between area and throughput. The throughput goal has been set to 1 Mbps in order to be comparable to other TRNGs proposals based on coherent sampling. This throughput threshold will set the lowest sampling frequency that can be used in our design.

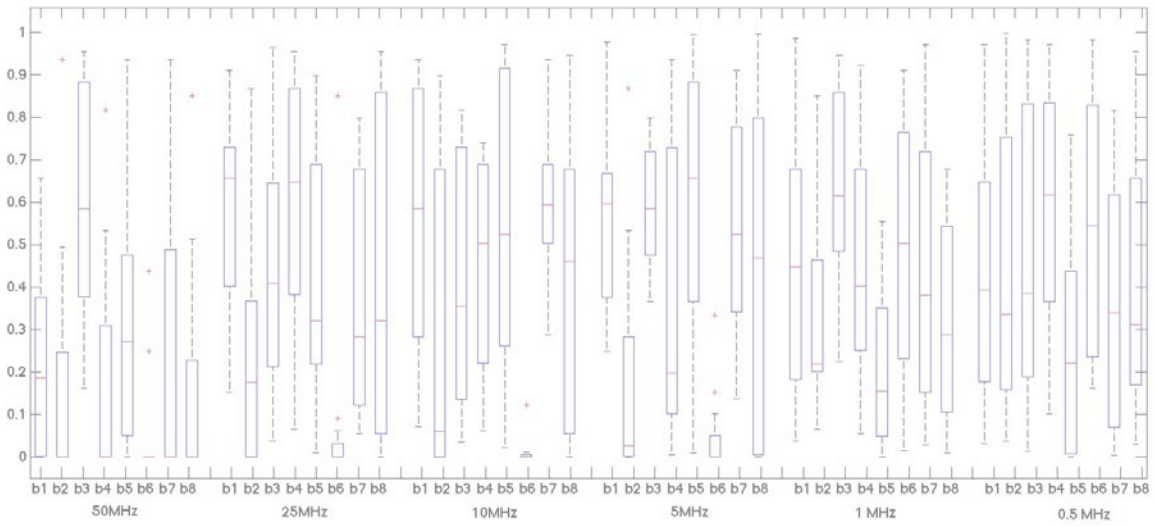


Figure 4.31: Boxplots of p-value distributions for each sampling stage ($b1$ to $b8$) and different frequencies.

4.3.2.1 Testing Randomness

The testing of our proposal has been carried out using the NIST statistical test suite [114], as commonly done to validate previous proposals (e.g., [54, 78, 37]). To transfer the bits generated by the TRNG in the FPGA to the host computer where the NIST tests are executed, a FIFO memory and a RS232 communication protocol have been used. In addition, the post-processing has been conducted in the host computer in order to reduce the acquisition time of the traces.

We have evaluated the TRNG output for the foloowing set of sampling frequencies: 50, 25, 4, 1, and 0.5 MHz). A higher sampling frequency will imply a higher throughput, but also a lower quality of the random bits due to the fact that the jitter accumulation time is shorter. According to the study presented in [61], a longer accumulation time is desirable so that the contribution of the thermal noise (responsible of the non-deterministic jitter) is perceptible. On the other hand, the use of a longer accumulation time causes that the flicker noise (responsible of the deterministic jitter) dominates the jitter. This paradox forces designers to find a trade-off to set the sampling frequency.

For the post-processing, we have tested the minimum parity filter order (bit-wise XOR tree) necessary to pass the NIST tests for the different sampling frequencies

studied. A 3^{rd} order filter is needed for 50 MHz, while a 2^{nd} order filter suffices for the rest. As expected, the post-processing necessities are higher when higher sampling frequencies are used. Although many sampling frequencies need the same order parity filter, it is important to notice that the proportion of failed tests before the post-processing rises when the sampling frequency is increased, as explained below. This is a crucial point if for some reasons the TRNG will be used without the post-processing block.

We have evaluated the quality of the raw data before the post-processing for the six sampling frequencies studied. Fig.4.31 shows boxplots of the p-value distribution for each sampling stage ($b1$ to $b8$) and different frequencies. According to the documentation provided by NIST, a random stream must present uniformity in the distribution of its p-values. It can be seen in Fig.4.31 that higher sampling frequencies presents less uniformity for its p-values distribution than lower sampling frequencies, which are more uniform.

Further evidence of this phenomenon is presented in Table 4.3, which shows the proportion of traces that pass the statistical tests (PR) and the average p-value (PV) for the different sampling stages and frequencies. Note that traces corresponding to $b5$ and $b6$ perform quite badly, specially $b6$. For sampling frequencies of 0.5 MHz and 1 MHz, only a single trace ($b5$) fails the NIST tests before the post-processing. It is noteworthy, however, that $b5$ fails the tests by a narrow margin. Three traces of $b5$ fail the tests for the sampling frequencies of 5 MHz, 10 MHz and 25 MHz, and 7 traces fail for 50 MHz. As for $b6$ traces, they fail badly for the sampling frequencies between 5 MHz and 50 MHz. This consistent behavior in $b6$ is mainly due to the fact that the synthesizer has placed the sampling stage that generates the $b6$ stream in a way that causes a huge delay between the sampling (S_{A6}) and the sampled (S_{B6}) signals. This problem could be solved using a manual placement and routing process. In fact, we have tested this: using manual placement and routing and setting the sampling frequency to 50 MHz results in a design such that the raw stream of bits without post-processing passes the NIST tests. Nevertheless, one major design goal of our proposal is to avoid such a manual procedures. We next show how the quality of the final output is not affected by sporadic low-quality stages.

Finally, we have evaluated the quality of our proposed TRNG after pre-processing. A sampling frequency of 1 MHz has been selected for this experimentation since

Table 4.4: Hardware results

	LUTs	Registers	Throughput (Mbps)
2 STRs	16	0	-
Sampling Circuit	8	32	-
Post-Processing	8	16	-
Total	32	48	4

Table 4.5: TRNG comparison

		Hardware Resources	Throughput	Hardware Complexity	Portability
Our proposal		32 LUTs 48 Registers	4 Mbps	medium	yes
Fischer et al. [54]		121 LCs 4 ESBs and 1 PLL	69 Kbps	medium	no
Kohl et al. [78]		12 LUTs 24 Registers	300 Kbps	high	yes
Valtchanov et al. [137]	RO-RO	15 LUTs 4 Registers	2 Mbps	high	yes
	RO-PLL	12 LCs 4 Registers and 1 PLL	2 Mbps	medium	no
	RO-DFS	11 LUTs 6 Registers and 2 DFS	2 Mbps	medium	no
Cherkaoui et al. [32]		320 LUTs 320 Registers	200 Mbps	medium	yes

this frequency offers a trade-of between throughput and randomness quality before the post-processing stage. We have opted for having a good quality signal without post-processing in order to make stronger our TRNG proposal against side channel attacks. ENT [142], DIEHARD [91], and NIST [114] suites have been used for analyzing the randomness quality.

In Table 4.6 we summarize the results obtained with ENT, which resemble those obtained with a genuine random variable: the chi-square test is passed; entropy is extremely high; the serial correlation is very low; etc. DIEHARD is a much more demanding battery of tests for checking randomness. As in the case of NIST tests, they are particularly designed for cryptographic applications. To show evidence that our proposed TRNG behaves as a random variable, in Fig. 4.32 we depict the distribution of p-values for all tests included in both suites. In particular, the p-values in both tests are in the interval between 0.2 and 0.8 and the TRNG passes both the NIST and DIEHARD batteries of tests. From all of the above, we can conclude that our proposed TRNG outputs a bit streams that looks like a true random variable.

4.3.2.2 Hardware results

The results presented in this subsection have been obtained for a Spartan-3E XC3S500E FPGA and correspond to our chosen design with a sampling frequency of 1 Mhz. The architecture consists of two 8-stage STRs, eight sampling stages,

Table 4.6: ENT results for a sampling frequency set to 1 MHz.

Entropy	7.999987
Compression	0%
Chi Square distribution	261.92 (36.96%)
Arithmetic mean	127.5110
Monte Carlo value for Pi	3.141718075
Serial correlation coefficient	0.000368

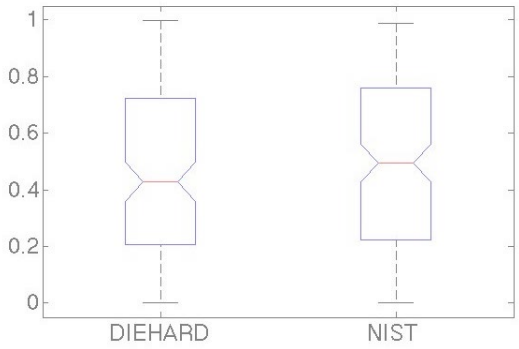


Figure 4.32: Distribution of p-values for DIEHARD and NIST test suites.

and a second order parity filter as post-processing block. Table 4.5 summarizes the resources needed to implement our TRNG and the final throughput obtained.

Since each STR stage uses a single LUT, the STRs occupies $2 \times L$ LUTs. As shown in Fig.4.28, the sampler structure uses 4 registers and an XOR gate (1 LUT). Therefore, the number of LUTs used by the sampler structure is L and the number of registers $4 \times L$. Finally, the post-processing necessities depend on the parity filter of order n. The LUTs used by the post-processing is also conditioned by the inputs of each LUT. Since a 4-input XOR gate, as in the case of 2-input XOR gates, can be implemented using 1 LUT, the number of LUTs and registers will be L and $n \times L$, respectively—the filter order is 2 for 1 MHz sampling frequency, as explained in Section 4.3.2.1.

In summary, observing the results above we can conclude that each raw random bit (before the post-processing) has a cost of 3 LUTs and 4 registers. Therefore, for a given sampling frequency ($f_{sampling}$), a designer could improve the throughput by adding more stages to the STRs. This will result in $f_{sampling}$ bps per additional stage. On the other hand, this improvement translates into a circuit area penalty of 3 LUTs and 4 registers per additional stage.

4.3.2.3 Comparison with others FPGA-based TRNGs

We finally present a comparison between our proposal and other TRNG designs that use coherent sampling. We also include the proposal of Cherkaoui et al. [32] since it is based on STRs. For each proposal, we have analyzed the hardware resources needed and the throughput offered. Besides, we have also considered the hardware complexity (including the degree of automation of the design) and its portability (device independence). Regarding hardware complexity, we distinguish among three categories: 1) *low complexity* is devoted for designs that can be implemented easily; 2) *medium complexity* implies designs that need to use hard-macros or specific components like PLL or DFS; finally 3) *high complexity* considers designs that require a manual place and route process. Finally, the portability column represents whether the design needs special resources or efforts to be implemented in different FPGA vendors or devices.

We emphasize here that the hardware results presented in Table 4.5 constitute an estimation for the designs in which the authors do not provide specific results. For Cherkaoui et al.'s proposal, we selected the architecture that implements 255 stages.

Table 4.5 shows the comparison between our design and other TRNG proposals. It can be noted that our proposal offers a very good trade-off among the set of parameters evaluated. TRNGs that need a complicated place and route process (e.g. [78] and RO-RO [137]) are superior in terms of hardware resources, but these designs have the drawback of requiring a specific design for each particular device. Among the TRNGs based on coherent sampling, our design offers the highest throughput. Note that this could be even better if a higher sampling frequency would have been selected. On the other hand, Cherkaoui et al.'s TRNG presents the highest throughput, but uses around 10 times more resources than our proposal. Finally, it is worth mentioning that our proposal is highly portable and complaint with the three requirements set in Section 4.3; lightweightness, robustness and portability.

4.3.3 Conclusions of our TRNG

In this section, we have proposed a TRNG based on coherent sampling, which is a phenomenon that seems to provide good results in previous proposals. Most previous works rely on either a Phase-Looked Loop or a Ring Oscillator. The use of these

components has one major drawback: it makes the design dependent on the FPGA vendor—e.g., not all FPGA vendors support PLLs—and requires manual placement and routing for setting particular frequencies for each device. To avoid these two drawbacks, we propose a novel design where ROs or PLLs are replaced by Self-timed Rings. We argue that the use of STRs is very convenient, for it provides robustness against frequency and voltage variations while, simultaneously, offers one independent source of entropy for each ring stage. Thus, the resulting TRNG combines the power of coherent sampling and the hardness and portability linked to STRs. Furthermore, our design does not depend on the FPGA vendor, and the placement and routing is performed automatically by the synthesis tool.

Our proposal outperforms all previous TRNGs based on STRs, and its throughput could be further increased if we relax our conditions about the quality of the random signals before the post-processing. We have studied in detail the most restrictive design with a sampling frequency set to 1 MHz. In terms of randomness, our TRNG passes all batteries of tests for checking the randomness of a random number generator (ENT and DIEHARD), and also others like NIST that are devoted to evaluate generators designed for cryptographic applications.

4.4 Conclusions

Lightweightness, robustness and portability are desired features for resource-constrained TRNGs. Motivated by these goals, many researchers have pointed out the convenience of using FPGAs as TRNG platforms, due to their low cost and versatility [50, 132, 144]. However, FPGAs offer a resource-constrained environment (fixed logic blocks) that does not include analog blocks, which are frequently employed to generate very entropic outputs.

In this chapter it has been presented an overview about TRNGs implemented on FPGAs. In the state of the art section 4.1 it has been reported the basics to design a TRNG. Moreover, a summary of the proposals contained in the literature has been presented. In addition, some attacks against TRNGs have been included.

In 4.2 it has been analysed a very promising TRNG designed by Cherkaoui et al. [32]. This TRNG is based on sampling several jittery signals, generated by an STR, and collecting them using a XOR-tree in order to obtain a random bit at the

output. A lightweight version (XOR-tree purely combinational), suitable for low cost applications, was evaluated in typical scenarios considered in TRNG evaluation. Temperature variation, underpowering, power glitches and clock glitches were taken into account. To the best of our knowledge, this is the first time that clock glitches have been considered in the evaluation of a TRNG. As conclusion of this analysis, some vulnerabilities in the lightweight version have been found. In addition, the original implementation has been tested obtaining good results in terms of security. We can conclude that the proposal presented by Cherkaoui et al. in [32] is secure if it is implemented correctly. On the other hand, it is not suitable for low cost applications like RFID.

Finally, due to the necessity of a secure lightweight TRNG in some applications, we have proposed a new TRNG based on the coherent sampling in 4.3. Our design is based on the idea presented by Kohlbrenner et al.. In order to overcome the weaknesses introduced by ROs, we have replaced ROs by STRs. STRs offer independent entropy sources of randomness that can be tuned accurately. Therefore, our TRNG combines the power of coherent sampling and the hardness and portability linked to STRs. Our proposed TRNG presents a trade-off between hardware resources and throughput offering a secure output that passes all batteries of tests ([114] [142] [91]).

5

Conclusions

5.1 Conclusions

RFID is one of the most promising identification technologies. The resources used in RFID tags are limited by two important factors: economical and technical. The first one is related to the necessity of a low production cost for the massive deployment of the technology. The technical factor is connected to the power requirements for operating passive RFID tags.

It is important for the massive deployment of the technology to have an standard that guarantees the same operation rules for RFID systems around the world. EPC-C1G2 is the most used standard in the industry. This standard describes the way of operation for UHF RFID systems. One of the most interesting specifications is the requirement of a pseudo-random/random number generator (RN16) in the tag to guarantee the security of the communications.

All in all, security and privacy are issues of concern for low-cost RFID systems nowadays. Due to the constrained-resources environment, typical secure cryptographic approximations can not be used in tags. For example, it is commonly assumed (see, e.g.,[111]) that no more than 4000 Gate Equivalents can be devoted to security functions. EPC-C1G2 tags support simultaneous read attempts up to 1500 tags/sec under ideal conditions. However, this rate can be five or ten times smaller (500-150 tags/sec) in real-world environments [21]. Therefore the number of clock cycles used per reading is upper-bounded by 670 clock cycles, assuming that the RFID chip operates at a clock frequency of 100 kHz. We take 500 clock cycles as reference value because this limit is less than the above mentioned value and has

been used in previous works [87, 94]. Furthermore, they should not consume more than 10 μW, as low-cost tags are passive and, therefore, must harvest their power supply from the reader signal. (See, for example, [20] for an elaborate motivation on the need for low-power designs.) When these constraints are compared with the approximate 8120 GE [51, 102] required by a standard hash function like SHA-1 (which is an essential building block for most security protocols), it becomes clear the need for schemes that can provide some minimum security services while requiring as few resources as possible. For that reason, lightweight cryptography plays a key role in RFID systems.

As aforesaid in the introduction, despite the numerous contributions reported in the literature about lightweight cryptography, there is a lack of proposals that tackle in a realistic way the multiple requirements imposed by the technology. Typically, most of them do not provide information about their implementation:

- Very theoretical contributions that do not provide any proof of their lightweightness are reported in the literature [122] [110].

- In many cases, arguments in favor of their lightweightness are based on the use of some operations that are generally considered inexpensive by the authors . However, these estimations are not always correct, and the implementation of some proposals greatly exceeds the area limit of 4K GE [73][97][34].

- In other cases, the design turns out to be not so lightweight because of factors such as the bit length of the variables, the need for additional memory blocks –which is usually missed in the analysis of resources–, and the overhead impossed by selection and control logic. These and other aspects often make the final gate count much higher than expected [34][110].

In this thesis lightweight cryptographic implementations for RFID systems have been studied from a realistic point of view. These are the major contributions regarding the two major objectives proposed in the Introduction section:

1. Studying the main lightweight cryptographic primitives, analysing their footprint area and suitability to be used in low-cost RFID tags.

- In chapter 2 it has been presented an study about the footprint area of elements commonly used in lightweight cryptography. The footprint area is one of the most important requirements because it is directly related to the tag cost. We have analysed various design elements and their area estimation depending on the complexity of their implementation. For low-complexity blocks, we have proposed a simple architecture, while for those with higher complexity we have studied and proposed several possible architectures. Authors could use this study to know what operations can include in their lightweight algorithms.
- With the information obtained from the study of elements used in lightweight cryptography, an area estimator for lightweight algorithms has been developed. This estimator takes into account the area devoted to the data-path and establishes a linear relation with the control logic (20 %). The estimator offers an upper-bound of the total footprint area. We have tested our estimator against a library containing 120 lightweight functions obtaining good results. Moreover, in order to provide a more accurate estimation of the area, other relations between the area of the data-path and the control logic have been studied. In comparison with the first estimator (20 %), these procedures offer a more accurate estimation, even though the new estimation cannot be considered anymore as an upper-bound for the total footprint area.

2. As the EPC standard establishes the necessity of an embedded Random number generator (RN16).

2.1. Designing and implementing a pseudo-random number generator compliant with the EPC-C1G2 standard, obtaining the main metrics (area, power consumption and throughput).

* Regarding the Pseudo random number generators, we have proposed two lightweight secure PRNGs known as AKARI. Several implementation architectures oriented to optimize some critical parameters have been proposed. The main metrics related to the technology requirements (area, power consumption and throughput) have been obtained. In order to provide more accurate numbers, a library that contains the layout of the used cells has been used, giving access to

very valuable information that is generally unavailable when using generic libraries.

* AKARI PRNGs have been integrated into two lightweight authentication protocols that comply with the standard EPC-C1G2. Both schemes rely on the use of a sufficiently good PRNG, but the particular choice is left to implementers. Given that such a component is critical to guarantee that the resulting circuit will fit a low-cost RFID tag, we have explored the integration of the two AKARI designs. As most PRNG-based EPC-C1G2 protocols follow a similar working scheme, we have designed an architecture for a generic EPC-C1G2 protocol and then particularized it for each implemented protocol. The main metrics have been presented for different configurations. In addition, the impact of the EPC module in a complete RFID tag has been studied .

2.2. Designing and implementing a True-random number generator, evaluating its suitability for low-cost RFID tags.

* Concerning True-random number generators, in chapter 4 it has been reported an overview about the state of the art of TRNGs. This overview includes main evaluation procedures, entropy extraction techniques and attacks. As a major contribution has been reported an analysis of a novel TRNG presented in [32]. This analysis includes typical scenarios used in the evaluation of TRNGs (temperature variation, underpowering and power glitches). In addition, the effects of clock glitches on a TRNG have been studied for the first time. It has been proved that a lightweight design like this is vulnerable against clock glitches attacks. Two solutions has been proposed in order to thwart these attacks . The first one includes a ripple structure in the XOR-tree (weak point of the TRNG) to reduce the critical path. The second one is oriented to devices that have an external clock pin. We have proposed to include a clock-observation circuit in order to ensure a glitch-free clock signal($clkgf$) for the D flip-flops of the XOR tree. Moreover, a general formal model to guide designers and evaluators to systematically test their architectures against this kind of attack has been presented.

* Finally, due to the necessity of a secure lightweight TRNG in some applications, we have proposed a new TRNG based on the coherent sampling technique in 4.3. Our design is based on the idea presented by Kohlbrenner et al. In order to overcome the weaknesses introduced by ROs, we have replaced ROs by STRs. STRs offer independent entropy sources of randomness that can be tuned accurately. Therefore, our TRNG combines the power of coherent sampling and the hardness and portability linked to STRs. Our proposed TRNG presents a trade-off between hardware resources and throughput offering a secure output that passes all batteries of tests ([114] [142] [91]). Evidence of the Gaussian jitter has been shown in **Fig.4.30**. Finally, a resource comparison with other TRNGs has been performed.

5.2 Future work

The work presented in this thesis can be extended in a number of ways.

Regarding chapter 2, the work related to the study of lightweight primitives and the estimator can be extended as follows:

- One natural direction for future work is the inclusion of other commonly used elements in the study, such as for example S-boxes of non-linear filters. Including these new blocks in the estimator.

- It would be interesting to extend our estimates to include other prominent parameters, primarily throughput and power consumption, as these have also significant influence in design choices.

- It would be very meaningful to compare the estimator results with the results provided by High level synthesis tools (e.g. Synphony HLS by Synopsys). High-Level Synthesis tools accelerates algorithm creation by enabling C, C++ and System C specifications to be directly targeted into an FPGA.

Concerning chapter 3, possible future lines in the PRNG area could be the followings:

- Based on AKARI PRNGs, designing a new PRNG that eliminates the last filter that penalizes the throughput using more complex operations.

True-number generators line (chapter 4) can be extended as follows:

- It could be interesting to implement in an ASIC the proposed TRNG in order to see how affects the surrounding conditions to the randomness.

- Other possible line is in the field of on-line test. As aforementioned, it is mandatory to guarantee a randomness level at the output. Designing lightweight on-line tests is a very challenging task in the future.

- In the field of fault attacks evaluation in TRNGs, there is a lack of formality in the evaluation of TRNGs against fault attacks. Create a generalized formulation would be essential to systematically evaluate TRNGs for different attacks.

Finally, designing and implementing our own authentication protocol for EPC-C1G2, where AKARI PRNGs and our TRNG could be integrated would be a very meaningful contribution in the field of low-cost RFID.

5.3 List of publications related to this thesis

5.3.1 Main thesis publications

J1 Martin, H.; Peris-Lopez, P.; Tapiador, J.E.; San Millan, E., "An Estimator for the ASIC Footprint Area of Lightweight Cryptographic Algorithms," Industrial Informatics, IEEE Transactions on , vol.10, no.2, pp.1216,1225, May 2014 doi: 10.1109/TII.2013.2288576 **IF=8,785**

J2 Martin, H.; San Millan, E.; Peris-Lopez, P.; Tapiador, J.E., "Efficient ASIC Implementation and Analysis of Two EPC-C1G2 RFID Authentication Protocols," Sensors Journal, IEEE , vol.13, no.10, pp.3537,3547, Oct. 2013 doi: 10.1109/JSEN.2013.2270404 **IF=1.852**

C1 Martin, H.; San Millan, E.; Entrena, L.; Lopez, P.P.; Castro, J.C.H., "AKARI-X: A pseudorandom number generator for secure lightweight systems," On-Line Testing Symposium (IOLTS), 2011 IEEE 17th International , vol., no., pp.228,233, 13-15 July 2011 doi: 10.1109/IOLTS.2011.5994534

J3 Martin, H.; Korak, T.; Millan, E.S.; Hutter, M., "Fault Attacks on STRNGs: Impact of Glitches, Temperature, and Underpowering on Randomness," Information Forensics and Security, IEEE Transactions on , vol.10, no.2, pp.266,277, Feb. 2015 doi: 10.1109/TIFS.2014.2374072 **IF=2.065**

J4 Martin, H.; Peris-Lopez, P.; Tapiador, J.E.; San Millan, E., "A New TRNG based on Coherent Sampling with Self-timed Rings" *Sent for review to the IEEE Transactions on Industrial Informatics* **IF=8,785**

5.3.2 Other contributions

C2 Martin, H.; San Millan, E.; Entrena, L.; Peris-Lopez, P., "Implementation of Secure Lightweight PRNGs for RFID." Conference on Design of Circuits and Integrated Systems (DCIS 2010).

C3 Martin, H.; Peris-Lopez, P.; San Millan, E.; Entrena, L.,"ATOM: A Pseudorandom Number Generator Suitable for RFID Protocols."Conference on Design of Circuits and Integrated Systems (DCIS 2011).

C4 Martin, H.; Vaskova, A; Lopez-Ongil, C.; San Millan, E.; Portela-Garcia, M., "Effect of ionizing radiation on TRNGs for safe telecommunications: Robustness and randomness," On-Line Testing Symposium (IOLTS), 2014 IEEE 20th International , vol., no., pp.202,205, 7-9 July 2014 doi: 10.1109/IOLTS.2014.6873697

C5 Gross, H.; Wenger, E.; Martin, H.; Hutter, M., PIONEER−a Prototype for the Internet of Things based on an Extendable EPC Gen2 RFID Tag. RFIDsec 2014 July 21−23, 2014, Oxford, UK.

J5 Martin, H.; Peris-Lopez, P.; Tapiador, J.E.; San Millan, E., Hardware Implementation of the Tav-128 Hash Function for RFID Systems *Sent for review to the Microelectronics Journal* **IF=0.924**

A

Data Set Functions

```
##
## Function names: F1, F2, ..., F30
##
## Symbols used:
##       N:  bit length of variables used
##       Z:  output value
##       Xi: input variables
##       Yi: intermediate variables
##
## There are:
##       10 functions with 2 input variables
##       10 functions with 4 input variables
##       10 functions with 6 input variables
##
## Each function should be implemented for N = 32, 64, 96, and 128 bits
     (120 different designs)
##
```

```
Z = F1(X1,X2)
For i=1 to N
        X1  =  X1 >> 3
        X2 = X1 >> 7
        X1  = (X1+X2)>3+ 0x789
```

```
Z= (X1+X2)>>7
```

```
Z = F2(X1,X2)
For i=1 to N
        X1= (X1+X2)  >> 3
        X2 = (X1 >> 7)+X2
        X1 = (X1 XOR X2)+ 0x789
        X2 = (X2 AND X1)+ 0x765 XOR X2
Z= (X1+X2)>>7
```

```
Z = F3(X1,X2)
For i=1 to N
        X1 = (X1 XOR X2) XOR (X1 AND X2)
        X2 = (X1 + X2)>>5 XOR (X1 + X2)
        X1 = (X2 >> 8) + X1
        X2 = (X1 >> 6) XOR X2
Z = (X1 XOR X2) AND (X1 OR X2)
```

```
Z = F4(X1,X2)
For i=1 to N
        Y1= (X1 + X2) >> 3
        X1 = X1 XOR (Y1>>1)
        X2 = X2 XOR X1+Y1
Z= (X1+X2)>>7 XOR Y1
```

```
Z = F5(X1,X2)
For i=1 to N
     Y1= (X1 + X2 >> 3) AND X1
     Y2= (X2 + Y1 >> 3) AND X1
     X1 = X1 XOR Y1
    X2 = X2 + Y2
Z=  (Y1 XOR Y2) XOR (X1 AND X2)
```

```
Z = F6(X1,X2,X3,X4)
For i=1 to N
```

```
        X1 = X1 XOR X2 XOR X3 XOR X4
        X2 = X2 XOR X1 + X3
        X3 = (X3 XOR X2 + X4)>>1
        X4 = X3 + X4 XOR 0x876
        X1 = X4 AND X2 XOR (X1>>5)
Z = X1 + X2 + X3 + X4
```

```
Z = F7(X1,X2,X3,X4)
For i=1 to N
        Y1 = X1 XOR X2 XOR X3 XOR X4
        X2 = X2 XOR X1 + Y1
        X3 = (X3 XOR Y1)>>5 + X4
        X4 = X3 AND X4 XOR 0x876
        X1 = X4 + X2 XOR (Y1>>5)
Z = X1 XOR X2 + X3 + X4+ (Y1>>5)
```

```
Z = F8(X1,X2,X3,X4)
For i=1 to N
    Y1 = (X1 + X2)>>5
    IF ((Y1>>(N-1))==1)
            X2 = X2+ X3 OR X4
    IF  ((Y2>>(N-1))==0)
            X3 = X1+ X4 OR X2
    X4= (X3 + X4 + X1)>>7+X2
    X1 = X1 XOR (Y1>>5) + X3
Z = X3+X2 XOR (X1 OR X2)+X4
```

```
Z = F9(X1,X2,X3,X4)
For i=1 to N
    X1 = X1 + X2 XOR X3
    IF  (X1 XOR X2 == X3)
            X3 = (X1+ X2)>>5+ X4
    IF  (X2 OR X3 == X4)
            X2 = X1 + X4 OR X2 XOR 0x678
    X3= X3+(X1 XOR X3 + X2) >> 3
    X2 = X2 XOR X3 OR X1
```

```
Z = X3+X2 + (X1 OR X2)  XOR 0x786
```

```
Z = F10(X1,X2,X3,X4)
For i=1 to N
    Y1 = X1 XOR X2
    Y2 = X2 XOR X4
    IF  ((Y1>>(N-1))==1)
          X2 = X2+ X3 OR Y1
    IF  ((Y2>>(N-1))==0)
          X3 = X1+ X4 OR Y2
    X3= (X3 + X4 + X1)>>7+X2
    X1 = X1 XOR Y1
    X4 = X4 + Y2 XOR X1
Z = X3+X2 XOR Y1 + Y2
```

```
Z = F11(X1,X2,X3,X4,X5,X6)
For i=1 to N
   X1 = X1 +X4+ 0x78
   X2 = X2 + X6 XOR (X4 AND X5)
   Y1 = X1 XOR (X2 >> 6)
   X3 = (X1+ X3 + Y1)>1+X5
   X4 = (X1+ X4 + Y1)>3 XOR X3
   X5 = (X1+ X5 + Y1)>5 + (X2>>5)
   X6 = (X1+ X6 + Y1)>7 XOR (X1 + 0x67)
Z= X1 + X2 + X3 + X4 + X5 + X6
```

```
Z = F12(X1,X2,X3,X4,X5,X6)
For i=1 to N
    Y1 = (X1 + X2)>1
    Y2 = (X3 XOR X4)
    Y3 = (X5 + X6)
    X3 = (X1+ Y1 + Y1)>1
    X4 = (X2+ Y2 + Y1)>3
    X5 = (X3+ Y3 + Y1)>5
    X6 = X6 XOR (Y1+ Y2 + Y3)>7
```

```
    X1 = X1 XOR Y1
    X2 = X2 + Y2
Z= X1 + X2 + X3 + X4 + X5 + X6
```

```
Z = F13(X1,X2,X3,X4,X5,X6)
    Y1 = X1 XOR X2
    X1 = X1 +X4
    X2 = X2 + X6
    X3 = (X1+ X3 + Y1)>1
    X4 = (X1+ X4 + Y1)>3
    X5 = (X1+ X5 + Y1)>5
    X6 = (X1+ X6 + Y1)>7
Z= X1 + X2 + X3 + X4 + X5 + X6
```

```
Z = F14(X1,X2,X3,X4,X5,X6)
        Y1 = X1 XOR X2
        IF ((Y1 >> N-1) == 1)
            X1 = X1 >> 3
       IF( (Y1 >> N-1) == 0)
            X2 = X1 >> 3
       X3 = (X1+ X3)>>6 + Y1
       X4 = (X2+ X4) XOR 0x77 + Y1
       X5 = (X1+ X5)>>5 + Y1
       X6 = (X1+ X6 + Y1)>7
Z= X1 + X2 + X3 + X4 + X5 + X6
```

```
Z = F15(X1,X2,X3,X4,X5,X6)
        Y1 = X1 + X2
        Y2 = X3 + X4
        X1 = X1 + Y2
        X2 = X2 + Y2
    IF ((X4 AND X5) == 0xF)
       X3 = X4 + X1 XOR X2
    X4 = X4+X1 XOR (Y1>>5)
    X5 = X5+X3 AND (Y2>>7)+X3
```

```
    X6 = X6>>6+Y1
Z= X1 XOR X2 + X3 + X4 XOR X5 + X6 XOR (Y1+Y2)
```

```
Z = F16 (X1, X2)
        Y1 = (X1 XOR X2) >> 16
        FOR i=1 TO N
                Y2 = (Y1 >> 16) OR X2
                Y2 = Y2 + 0x3B2FA064
                X2 = X2 XOR Y2
        END-FOR
        Z = Y2 + X2
```

```
Z = F17 (X1, X2)
        Y1 = (X1 + X2) >> 16
        IF (Y1 == 0)
                X1 = X1 >> 16
                X2 = X2 >> 16
        END-IF
        FOR i=1 TO N
                Y1 = (Y1 + X1) >> 8
                Y2 = (Y1 + X2) >> 8
                X1 = X2 XOR Y2
                X2 = X1 XOR Y1
        END-FOR
        Z = X1 OR X2
```

```
Z = F18 (X1, X2)
        Y1 = (X1 XOR X2) + (X1 AND X2)
        Y2 = (X1 AND X2) + (X1 OR X2)
        Y3 = (Y1 XOR Y2) >> 16
        IF (Y1 == 0)
                Y3 = Y3 + 0X17D4A0BB
        END-IF
        IF (Y2 AND Y3 == 0)
                Y3 = Y3 XOR X1
```

```
        END–IF
        Z = Y1 + Y2 + Y3
```

```
Z = F19 (X1, X2)
        Y1 = 0
        FOR i=1 TO N
                Y1 = Y1 + ( (X1 >> 8) XOR X2 )
        END–FOR
        Z = Y1 XOR X1 XOR X2
```

```
Z = F20 (X1, X2)
        FOR i=1 TO N
                Y1 = (X1 >> 32) + X2
                IF (Y1 == 0)
                        X2 = X2 >> 8
                END–IF
        END–FOR
        Z = Y1 AND X2
```

```
Z = F21 (X1, X2, X3, X4)
        Y1 = X1 XOR X2 XOR X3 XOR X4
        FOR i=1 TO 32
                Y2 = (X1 + X2 + X3 + X4) >> 16
                Y3 = (Y1 XOR Y2) + (Y1 AND Y2)
                X1 = (X1 >> 1) XOR Y3
                X2 = (X2 >> 1) XOR Y2
                X3 = (X3 >> 1) XOR Y3
                X4 = (X4 >> 1) XOR Y2
        END–FOR
        Z = Y2 XOR Y3
```

```
Z = F22 (X1, X2, X3, X4)
        Y1 = (X1 AND X2) + (X3 OR X4)
```

```
        Y2 = (X1 XOR X2 XOR X3 XOR X4) >> 16
        Y3 = (X1 + Y1) XOR X1
        Y4 = (X2 + Y2) OR X2
        Y5 = (X3 + Y1) XOR X3
        Y6 = (X4 + Y2) OR X4
        Z = Y1 XOR Y2 XOR Y3 XOR Y4 XOR Y5 XOR Y6
```

```
Z = F23 (X1, X2, X3, X4)
        Y1 = 0
        FOR i=1 TO 16
                X1 = (X1 >> 8) XOR X2
                X2 = (X2 >> 8) XOR X3
                X3 = (X3 >> 8) XOR X4
                X4 = (X4 >> 8) XOR X1
                Y1 = Y1 XOR ( (X1 XOR X2) + (X3 XOR X4) )
        END-FOR
        Z = Y1
```

```
Z = F24 (X1, X2, X3, X4)
        Y3 = 0
        IF (X1 AND X2 = 0)
                FOR i=1 TO N
                        Y1 = (Y1 >> 8) XOR X1
                        Y2 = (Y1 >> 8) XOR X2
                        Y3 = Y3 + (Y1 AND Y2)
                END-FOR
        END-IF
        Y4 = X3
        Y5 = X4
        FOR i=1 TO N
                Y3 = Y3 XOR Y4 XOR Y5
                Y4 = (Y4 >> 8) + Y3
                Y5 = (Y5 >> 8) + Y3
        END-FOR
        Z = Y3 XOR X1 XOR X2 XOR X3 XOR X4
```

```
Z = F25 (X1, X2, X3, X4)
        Y1 = 0
        FOR i=1 TO N
                Y1 = Y1 + (X1 XOR x2)
                X1 = (X1 >> 16) AND Y1
                X2 = (X2 OR Y1) AND X1
        END-FOR
        IF (X3 XOR X4 == 0)
                Y1 = (Y1 >> 16) + X3 + X4
                X3 = X3 XOR Y1
                X4 = X4 XOR Y1
        END-IF
        Z = Y1 XOR X3 XOR X4
```

```
Z = F26 (X1, X2, X3, X4, X5, X6)
        Y1 = (X1 + X2) XOR X5
        Y2 = (X3 + X4) XOR X6
        IF (Y1 AND Y2 == 0)
                Y1 = Y1 >> 16
                Y2 = Y2 >> 16
        END-IF
        FOR i=1 TO N
                Y1 = Y1 XOR (X5 AND X6)
                Y2 = Y1 OR (X1 + X6) OR 0x0F0F0F0F
                X5 = X5 XOR (X1 + Y1 + Y2)
                X6 = X6 AND (X1 + Y1 + Y2)
        END-FOR
        Z = Y1 XOR Y2
```

```
Z = F27 (X1, X2, X3, X4, X5, X6)
        Y1 = 0
        FOR i=1 TO 64
                Y1 = X1 XOR X2 XOR X3
                X1 = (X4 >> 8) + Y1
                X2 = (X5 >> 16) + Y1
```

```
                X3 = (X6 >> 24) + Y1
        END-FOR
        Z = Y1 AND (X4 XOR X5 XOR X6)
```

```
Z = F28 (X1, X2, X3, X4, X5, X6)
        Y1 = (X1 + X2) >> 16
        Y2 = (X3 + X4 + X5) >> 8
        Y3 = (X6 AND Y1) + (X6 XOR Y2)
        IF ( Y1 XOR Y2 XOR Y3 == 0)
                Y1 = Y1 + 0XF1F1F1F1
                Y2 = (Y1 OR Y2) + (Y1 AND Y3)
        END-IF
        Z = Y3 XOR (Y1 + Y2)
```

```
Z = F29 (X1, X2, X3, X4, X5, X6)
        FOR i=1 TO 16
                X1 = X1 XOR X4
                X2 = X2 XOR X5
                X3 = X3 XOR X6
                Y1 = X1 + X2 + X3
        END-FOR
        Y1 = (Y1 >> 16) + (X1 XOR X2 XOR X3)
        Z = (Y1 XOR X4) + (Y1 XOR X5) + (Y1 XOR X6)
```

```
Z = F30 (X1, X2, X3, X4, X5, X6)
        Y1 = X1 XOR X2 XOR X3 XOR X4 XOR X5 XOR X6
        FOR i=1 TO 16
                X1 = (X1 >> 8) + Y1
                X2 = (X2 >> 8) + Y1
                X3 = (X3 >> 8) + Y1
                X4 = (X4 >> 8) + Y1
                X5 = (X5 >> 8) + Y1
                X6 = (X6 >> 8) + Y1
                Y1 = X1 XOR X2 XOR X3 XOR X4 XOR X5 XOR X6
        END-FOR
```

```
Z = 0X0F0F0F0F AND Y1
```

References

[1] Amis semiconductor c035u cmos. *Design Rules. Rev. A*, 2006.

[2] Synopsys design compiler user guide. *Version D-2010.03-SP2*, 2010.

[3] ISO/IEC 18000-6. Information technology – radio frequency identification for item management – part 6: Parameters for air interface communications at 860 mhz to 960 mhz. 2013.

[4] A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, and C. Vikkelsoe. Present: An ultra-lightweight block cipher. In proceedings of CHES 2007, 2007.

[5] A. Juels, D. Molner, and D. Wagner, . Security and privacy issues in epassports. Proc. First Int'l Conf. Security and Privacy for Emerging Areas in Comm. Networks, 2005.

[6] Michel Agoyan, Jean-Max Dutertre, David Naccache, Bruno Robisson, and Assia Tria. When Clocks Fail: On Critical Paths and Clock Faults. In *Smart Card Research and Advanced Application*, pages 182–193. Springer, 2010.

[7] S.A. Ahson and M. Ilyas. *RFID Handbook: Applications, Technology, Security, and Privacy.* Taylor & Francis, 2008.

[8] J.A.A. Angulo, E. Kussener, H. Barthelemy, and B. Duval. A new oscillator-based random number generator. 2012. cited By (since 1996)0.

[9] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787–2805, 2010.

[10] Gildas Avoine, Xavier Carpent, and Benjamin Martin 0002. Privacy-friendly synchronized ultralightweight authentication protocols in the storm. *J. Network and Computer Applications*, 35(2):826–843, 2012.

[11] B. M. Baas. A low-power, high-performance, 1024-point FFT processor. *IEEE Journal of Solid-State Circuits*, 34(3):380–387, 1999.

[12] Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall, and Claire Whelan. The sorcerer's apprentice guide to fault attacks. *IACR Cryptology ePrint Archive*, 2004:100, 2004.

[13] S.G. Baskir and B. Ors. Implementation of a secure rfid protocol. In *Signal Processing and Communications Applications Conference (SIU), 2013 21st*, pages 1–4, April 2013.

[14] Pierre Bayon, Lilian Bossuet, Alain Aubert, Viktor Fischer, Francois Poucheret, Bruno Robisson, and Philippe Maurine. Contactless electromagnetic active attack on ring oscillator based true random number generator. In *Constructive Side-Channel Analysis and Secure Design - Third International Workshop, COSADE 2012, Darmstadt, Germany, May 3-4, 2012. Proceedings*, pages 151–166, 2012.

[15] R. Ben Atitallah, E. Senn, D. Chillet, M. Lanoe, and D. Blouin. An efficient framework for power-aware design of heterogeneous mpsoc. *Industrial Informatics, IEEE Transactions on*, 9(1):487–501, Feb 2013.

[16] F. Bernard, V. Fischer, and B. Valtchanov. Mathematical model of physical rngs based on coherent sampling. *Tatra Mountains - Mathematical Publications*, 45:1–14, 2010.

[17] L Blum, M Blum, and M Shub. A simple unpredictable pseudo random number generator. *SIAM J. Comput.*, 15(2):364–383, May 1986.

[18] N. Bochard, F. Bernard, and V. Fischer. Observing the randomness in ro-based trng. In *Reconfigurable Computing and FPGAs, 2009. ReConFig '09. International Conference on*, pages 237–242, Dec 2009.

[19] E. Bohl, M. Lewis, and S. Gallein. A true random number generator with on-line testability. 2014. cited By (since 1996)0.

[20] D. Brenk, J. Essel, J. Heidrich, R. Agethen, D. Kissinger, G. Hofer, G. Holweg, G. Fischer, and R. Weigel. Energy-efficient wireless sensing using a generic adc sensor interface within a passive multi-standard rfid transponder. *IEEE Sensors Journal*, 11(11):2698–2710, Nov. 2011.

[21] M. Brown, E. Zeisel, and R. Sabella. Chapter 2 - rfid tags. In *RFID+ Exam Cram*. Que, 2006.

[22] M. Burmester and J. Munilla. Lightweight RFID authentication with forward and backward security. *ACM Trans. Inf. Syst. Secur.*, 14(1):11:1–11:26, June 2011.

[23] Levente Buttyan and Jean-Pierre Hubaux. *Security and Cooperation in Wireless Networks: Thwarting Malicious and Selfish Behavior in the Age of Ubiquitous Computing*. Cambridge University Press, New York, NY, USA, 2007.

[24] Rafael Boix Carpi, Stjepan Picek, Lejla Batina, Federico Menarini, Domagoj Jakobovic, and Marin Golub. Glitch it if You Can: Novel Parameter Search Strategies for Successful Fault Injection. In Pankaj Rohatgi and Aurelien Francillon, editors, *Smart Card Research and Advanced Applications -? CARDIS 2013, 12th International Conference, Berlin, Germany, November 27-?29, 2013, Proceedings.*, LNCS. Springer, 2013.

[25] T.J. Chaney and C.E. Molnar. Anomalous behavior of synchronizer and arbiter circuits. *Computers, IEEE Transactions on*, C-22(4):421–422, April 1973.

[26] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template Attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2003.

[27] W. Che, H. Deng, X. Tan, and J. Wang. A random number generator for application in rfid tags. In *Networked RFID Systems and Lightweight Cryptography*, pages 279–287. Springer-Verlag, 2008.

[28] Wenyi Che, Huan Deng, Wang Tan, and Junyu Wang. A random number generator for application in rfid tags. In Peter H. Cole and Damith C. Ranasinghe, editors, *Networked RFID Systems and Lightweight Cryptography*, pages 279–287. Springer Berlin Heidelberg, 2008.

[29] W. Chen, W. Che, N. Yan, X. Tan, and H. Min. Ultra-low power truly random number generator for rfid tag. *Wireless Personal Communications*, 59(1):85–94, 2011. cited By (since 1996)0.

[30] Abdelkarim Cherkaoui, Viktor Fischer, Alain Aubert, and Laurent Fesquet. Comparison of self-timed ring and inverter ring oscillators as entropy sources in fpgas. In *DATE*, pages 1325–1330, 2012.

[31] Abdelkarim Cherkaoui, Viktor Fischer, Alain Aubert, and Laurent Fesquet. A self-timed ring based true random number generator. In *ASYNC*, pages 99–106, 2013.

[32] Abdelkarim Cherkaoui, Viktor Fischer, Laurent Fesquet, and Alain Aubert. A Very High Speed True Random Number Generator with Entropy Assessment. In Guido Bertoni and Jean-Sebastien Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013*, volume 8086 of *Lecture Notes in Computer Science*, pages 179–196. Springer Berlin Heidelberg, 2013.

[33] H.-Y. Chien and C.-W. Huang. A lightweight authentication protocol for low-cost RFID. *Journal of Signal Processing Systems*, 59(1):95–102, April 2010.

[34] H.Y. Chien. SASI: A new ultralightweight RFID authentication protocol providing strong authentication and strong integrity. *IEEE Transactions on Dependable and Secure Computing*, 4(4):337–340, Oct.-Dec. 2007.

[35] Peter H. Cole and Damith C. Ranasinghe. *Networked RFID Systems and Lightweight Cryptography: Raising Barriers to Product Counterfeiting*. 1 edition.

[36] National Intelligence Council. Disruptive civil technologies: Six technologies with potential impacts on us interests out to 2025. Conference Report - CR 2008-07, April 2008.

[37] O. Cret, A. Suciu, and T. Gyorfi. Practical issues in implementing trngs in fpgas based on the ring oscillator sampling method. In *10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC '08)*, pages 433–438. IEEE Computer Society, 2008.

[38] J.-L. Danger, S. Guilley, and P. Hoogvorst. High speed true random number generator based on open loop structures in {FPGAs}. *Microelectronics Journal*, 40(11):1650 – 1656, 2009.

[39] Paolo D'Arco and Alfredo De Santis. On ultralightweight rfid authentication protocols. *IEEE Trans. Dependable Sec. Comput.*, 8(4):548–563, 2011.

[40] Brian Panchalingam Mukunthan Stratis Chris Dargan, Gaurav Johnson. The use of radio frequency identification as a replacement for traditional barcoding. 2004.

[41] Robert B Davies. Exclusive or (xor) and hardware random number generators. *Website*, 2002.

[42] Markus Dichtl. How to predict the output of a hardware random number generator. In ColinD. Walter, CetinK. Koc, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 181–188. Springer Berlin Heidelberg, 2003.

[43] Markus Dichtl and Jovan Dj. Golic. High-speed true random number generation with logic gates only. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 45–62. Springer, 2007.

[44] D. M. Dobkin. Chapter 3 - Radio basics for UHF RFID. In *The RF in RFID*, pages 51–101. Newnes, 2008.

[45] O. Elissati, E. Yahya, S. Rieubon, and L. Fesquet. A novel high-speed multi-phase oscillator using self-timed rings. In *International Conference on Microelectronics (ICM'10)*, pages 204–207, 2010.

[46] Sho Endo, Takeshi Sugawara, Naofumi Homma, Takafumi Aoki, and Akashi Satoh. An on-chip glitchy-clock generator and its applicataion to sage-error attack. In *Second International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE 2011), 24-25 February 2011, Darmstadt, Germany*, Workshop Proceedings COSADE 2011, pages 175–182, 2011.

[47] EPCglobal Inc. Class-1 Generation-2 UHF RFID protocol for communications at 860 MHz - 960 MHz (version 1.2.0), 2008.

[48] F. Thornton, B. Haines, A. M. Das, H. Bhargava, A. Campbell, J. Kleinschmidt. Rfid security, protect the supply chain. Ed: Syngress Publishing INC. ISBN:1-59749-047-4, 2006.

[49] Benjamin Fabian, Tatiana Ermakova, and Cristian Muller. Shardis: A privacy-enhanced discovery service for rfid-based product information. *IEEE Trans. Industrial Informatics*, 8(3):707–718, 2012.

[50] X. Fang, Q. Wang, C. Guyeux, and J. M. Bahi. {FPGA} acceleration of a pseudorandom number generator based on chaotic iterations. *Journal of Information Security and Applications*, 19(1):78 – 87, 2014.

[51] M. Feldhofer and C. Rechberger. A case against currently used hash functions in rfid protocols. In *International Conference on the Move to Meaningful Internet Systems - OTM Workshops*, volume 4277 of *LNCS*, pages 372–381. Springer-Verlag, 2006.

[52] Klaus Finkenzeller. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification.* John Wiley & Sons, Inc., New York, NY, USA, 2 edition, 2003.

[53] FIPS. *Security Requirements for Cryptographic Modules*, May 2001.

[54] M. Fischer, V. Drutarovsky. True random number generator embedded in reconfigurable hardware. In *Proceedings of International Workshop on Cryptographic Hardware and Embedded Systems (CHES'02)*, volume 2523 of *Lecture Notes in Computer Science*, pages 415–430. Springer-Verlag, 2002.

[55] Viktor Fischer, Milos Drutarovsky, Martin Simka, Frederic Celle, and Universite Jean Monnet. A simple pll-based true random number generator for embedded digital systems. *Computing and Informatics*, pages 5–6.

[56] ISO: Internacional Organization for Standardization. http://www.iso.org/.

[57] B. Glover and H. Bhatt. *RFID Essentials (Theory in Practice)*. O'Reilly Media, Inc., 2006.

[58] T. Guneysu and C. Paar. Transforming write collisions in block rams into security applications. In *Field-Programmable Technology, 2009. FPT 2009. International Conference on*, pages 128–134, Dec 2009.

[59] Xu Guo, Meeta Srivastav, Sinan Huang, Dinesh Ganta, Michael B. Henry, Leyla Nazhandali, and Patrick Schaumont. Asic implementations of five sha-3 finalists. In Wolfgang Rosenstiel and Lothar Thiele, editors, *DATE*, pages 1006–1011. IEEE, 2012.

[60] T. Gyorfi, O. Cret, and A Suciu. High performance true random number generator based on fpga block rams. In *Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–8, May 2009.

[61] P. Haddad, Y. Teglia, F. Bernard, and V. Fischer. On the assumption of mutual independence of jitter realizations in p-trng stochastic models. In *Design, Automation and Test in Europe Conference and Exhibition (DATE'14)*, pages 1–6. IEEE Computer Society, 2014.

[62] S. Halevi. Cryptographic hash functions and their many applications. *USENIX Security Symposium*, 2009.

[63] Hamid Jabbar, Taikyeong Ted. Jeong. Rfid system integration. http://sciyo.com/download/pdf/pdfs_id/8487?...39k13d81v7ljriispot11toiq.

[64] J.C. Hernandez-Castro, J.M. Estevez-Tapiador, A. Ribagorda-Garnacho, and B. Ramos-Alvarez. Wheedham: an automatically designed block cipher by means of genetic programming. In *IEEE Congress on Evolutionary Computation*, pages 192–199, 2006.

[65] D.E. Holcomb, W.P. Burleson, and K. Fu. Power-up sram state as an identifying fingerprint and source of true random numbers. *Computers, IEEE Transactions on*, 58(9):1198–1210, 2009.

[66] S. Hosseini.-Khayat. A lightweight security protocol for ultra-low power asic implementation for wireless implantable medical devices. In *Medical Information Communication Technology (ISMICT), 2011 5th International Symposium on*, pages 6–9, March 2011.

[67] http://www.afcea.org. Radio frequency identification ready to deliver, armed forces communications and electronics association. January 2005.

[68] Miaoqing Huang, Kris Gaj, and Tarek A. El.-Ghazawi. New hardware architectures for montgomery modular multiplication algorithm. *IEEE Trans. Computers*, 60(7):923–936, 2011.

[69] Y.-J. Huang, W.-C. Lin, and H.-L. Li. Efficient implementation of RFID mutual authentication protocol. *IEEE Transactions on Industrial Electronics*, 59(12):4784–4791, Dec. 2012.

[70] Yu.Jung Huang, Ching.Chien Yuan, Ming.Kun Chen, Wei.Cheng Lin, and Hsien.Chiao Teng. Hardware implementation of rfid mutual authentication protocol. *Industrial Electronics, IEEE Transactions on*, 57(5):1573–1582, May 2010.

[71] Michael Hutter and Jörn-Marc Schmidt. The Temperature Side-Channel and Heating Fault Attacks. In Pankaj Rohatgi and Aurelien Francillon, editors, *Smart Card Research and Advanced Applications - CARDIS 2013, 12th International Conference, Berlin, Germany, November 27-29, 2013, Proceedings*. Springer, 2013.

[72] International Air Transport Association. Rfid business case for baggage tagging. http://www.iata.org/whatwedo/stb/Documents/RFID0case

[73] Guiyue Jin, Jiyu Jin, Xueheng Tao, and Baoying Li. A full scale authentication protocol for rfid conforming to epc class1 gen2 standard. In *Services Computing Conference (APSCC), 2012 IEEE Asia-Pacific*, pages 286–290, Dec 2012.

[74] A. Klimov and A. Shamir. A new class of invertible mappings. In *International Workshop on Cryptographic Hardware and Embedded Systems*, volume 2523 of *LNCS*, pages 471–484. Springer-Verlag, 2002.

[75] A. Klimov and A. Shamir. New applications of T-functions in block ciphers and hash functions. In *Fast Software Encryption*, volume 3557 of *LNCS*, pages 18–31. Springer Berlin Heidelberg, 2005.

[76] Alexander Klimov and Adi Shamir. A new class of invertible mappings. In Burton S. Kaliski Jr., Cetin Kaya Koc, and Christof Paar, editors, *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 470–483. Springer, 2002.

[77] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In Michael Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[78] Paul Kohlbrenner and Kris Gaj. An embedded true random number generator for fpgas. In *Proceedings of the 12th International Symposium on Field Programmable Gate Arrays (ACM/SIGDA'04)*, FPGA '04, pages 71–78, New York, NY, USA, 2004. ACM.

[79] Kwak, M., Kim, J. y Kim, K. Desynchronization and cloning resistant lightweight rfid authentication protocolo using integer arithmetic for low-cost tags. http://caislab.kaist.ac.kr/publication/paper_files/2009/SCIS2009_Minhea.pdf.

[80] Siew-Hwee Kwok, Yen-Ling Ee, Guanhan Chew, Kanghong Zheng, Khoongming Khoo, and Chik How Tan. A comparison of post-processing techniques for biased random number generators. In Claudio Agostino Ardagna and Jianying Zhou, editors, *WISTP*, volume 6633 of *Lecture Notes in Computer Science*, pages 175–190. Springer, 2011.

[81] Shuenn.-Yuh Lee, Liang.-Hung Wang, and Qiang Fang. A low-power rfid integrated circuits for intelligent healthcare systems. *Information Technology in Biomedicine, IEEE Transactions on*, 14(6):1387–1396, Nov 2010.

[82] Yang Li, Kazuo Ohta, and Kazuo Sakiyama. New fault-based side-channel attack using fault sensitivity. *IEEE Transactions on Information Forensics and Security*, 7(1):88–97, 2012.

[83] Zhaoyu Liu and Dichao Peng. True random number generator in rfid systems against traceability. In *Consumer Communications and Networking Conference, 2006. CCNC 2006. 3rd IEEE*, volume 1, pages 620–624, Jan 2006.

[84] F. Lozach, M. Ben-Romdhane, T. Graba, and J.L. Danger. Fpga design of an open-loop true random number generator. pages 615–622, 2013. cited By (since 1996)0.

[85] D. Lubicz and N. Bochard. Towards an oscillator based trng with a certified entropy rate. *Computers, IEEE Transactions on*, PP(99):1–1, 2014.

[86] Adnan Abu Mahfouz and Gerhard P. Hancke. Distance bounding: A practical security solution for real-time location systems. *IEEE Trans. Industrial Informatics*, 9(1):16–27, 2013.

[87] K. Mandal, X. Fan, and G. Gong. Warbler: A lightweight pseudorandom number generator for EPC C1 Gen2 tags. In *Radio Frequency Identification System Security*, volume 8 of *Cryptology and Information Security*, pages 73–84. 2012.

[88] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks – Revealing the Secrets of Smart Cards*. Springer, 2007. ISBN 978-0-387-30857-9.

[89] Manish Bhuptani, Shaaram Moradpour. Rfid field guide. Prentice Hall PTR Upper Saddle River, NJ, USA, 2005.

[90] A. Theodore Markettos and Simon W. Moore. The frequency injection attack on ring-oscillator-based true random number generators. In *Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '09, pages 317–331, Berlin, Heidelberg, 2009. Springer-Verlag.

[91] G. Marsaglia. The marsaglia random number cdrom including the diehard battery of tests of randomness. 1996.

[92] George Marsaglia and Wai Wan Tsang. Some difficult-to-pass tests of randomness. *Journal of Statistical Software*, 7(3):1–9, 1 2002.

[93] H. Martin, E. San Millan, L. Entrena, P. Peris-Lopez, and J. C. Hernandez-Castro. AKARI-X: A pseudorandom number generator for secure lightweight systems. In *IEEE International On-Line Testing Symposium (IOLTS)*, pages 228–233. IEEE Society, 2011.

[94] J. Melia.-Segui, Joaquin Garcia-Alfaro, and Jordi Herrera.-Joancomarti. Analysis and improvement of a pseudorandom number generator for epc gen2 tags. In Radu Sion, Reza Curtmola, Sven Dietrich, Aggelos Kiayias, Josep M. Miret, Kazue Sako, and Francesc Sebe, editors, *Financial Cryptography Workshops*, volume 6054 of *Lecture Notes in Computer Science*, pages 34–46. Springer, 2010.

[95] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996.

[96] David Molnar and David Wagner. Privacy and security in library rfid: Issues, practices, and architectures. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, CCS '04, pages 210–219, New York, NY, USA, 2004. ACM.

[97] M.M. Morshed, A. Atkins, and Hongnian Yu. An efficient and secure authentication protocol for rfid systems. In *Automation and Computing (ICAC), 2011 17th International Conference on*, pages 51–56, Sept 2011.

[98] Nadia Nedjah and Luiza de Macedo Mourelle. A review of modular multiplication methods ands respective hardware implementation. *Informatica (Slovenia)*, 30(1):111–129, 2006.

[99] Mundo NFC. Diferencia entre nfc y rfid. http://mundonfc.wordpress.com/2012/02/08/diferencia-entre-nfc-y-rfid/, 2012.

[100] NxP. UCODE EPC Gen2, Consulted on March 2013.

[101] Graz University of Technology. *UHF RFID Demo Tag.* [Online] (Last access Sept. 2014), Available at http://jce.iaik.tugraz.at/sic/Products/.

[102] Maire O'Neill. Low-cost SHA-1 hash function architecture for RFID tags. In *Workshop on RFID Security*, pages 41–51. Dominikus, S., Aigner, M. (eds.), 2008.

[103] P. Peris Lopez. Phd:lightweight cryptography in radio frequency identification (rfid) systems. http://www.lightweightcryptography.com/, 2008.

[104] Pedro Peris-Lopez, Julio Cesar Hernandez-Castro, Juan M. Estevez-Tapiador, and Arturo Ribagorda. An efficient authentication protocol in rfid systems resistant to active attacks. In EUC Conferences and Workshop: SecUbiq Workshop,, 2007.

[105] Pedro Peris.-Lopez, Julio Cesar Hernandez Castro, Juan M. Estevez.-Tapiador, and Arturo Ribagorda. Lamed - a prng for epc class-1 generation-2 rfid specification. *Computer Standards & Interfaces*, 31(1):88–97, 2009.

[106] Selwyn Piramuthu. Rfid mutual authentication protocols. *Decision Support Systems*, 50(2):387–393, 2011.

[107] Thomas Plos, Michael Hutter, Martin Feldhofer, M. Stiglic, and F. Cavaliere. Security-enabled near-field communication tag with flexible architecture supporting asymmetric cryptography. *IEEE Trans. VLSI Syst.*, 21(11):1965–1974, 2013.

[108] R. A. Potyrailo, D. Monk, W. G. Morris, S. Klensmeden, H. Ehring, T. Wortley, V. Pizzi, J. Carter, and G. Gach. Integration of passive multivariable RFID sensors into single-use biopharmaceutical manufacturing components. In *IEEE International Conference on RFID*, pages 1–7, April 2010.

[109] R.A. Potyrailo, C. Surman, W.G. Morris, T. Wortley, M. Vincent, R. Diana, V. Pizzi, J. Carter, and G. Gach. Lab-scale long-term operation of passive multivariable rfid temperature sensors integrated into single-use bioprocess components. In *IEEE International Conference on RFID-Technologies and Applications (RFID-TA)*, pages 16–19, Sept.

[110] N. Rama and R. Suganya. Ssl-map: A more secure gossamer-based mutual authentication protocol for passive rfid tags. In *International Journal on Computer Science and Engineering*, pages 363–367, 2010.

[111] D. Ranasinghe, D. Engels, and P. Cole. Low-cost RFID systems: confronting security and privacy. In *Proceedings of Auto-ID Labs Research Workshop*, 2004.

[112] Behzad Razavi. *Fundamentals of microelectronics*. Wiley, Hoboken, NJ, 2008.

[113] H. Reinisch, M. Wiessflecker, S. Gruber, H. Unterassinger, G. Hofer, M. Klamminger, W. Pribyl, and G. Holweg. A multifrequency passive sensing tag with on-chip temperature sensor and off-chip sensor interface using EPC HF and UHF RFID technology. *IEEE Journal of Solid-State Circuits*, 46(12):3075–3088, Dec. 2011.

[114] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo. A statistical test suite for random and pseudorandom number generators for cryptographic applications. *Technical Report*, 2001.

[115] S. Kinoshita, M. Ohkubo, F. Hoshino, G. Morohashi, O. Shionoiri, and A. Kanai,. Privacy enhanced active rfid tag. Proc. Int'l Workshop Exploiting Context Histories in Smart Environments, 2005.

[116] S.S. Saab, J.G. Hobeika, and IE. Ouaiss. A novel pseudorandom noise and band jammer generator using a composite sinusoidal function. *IEEE Transactions on Signal Processing*, 58(2):535–543, Feb 2010.

[117] A. P. Sample, D. J. Yeager, P. S. Powledge, A. V. Mamishev, and J. R. Smith. Design of an RFID-based battery-free programmable sensing platform. *IEEE Transactions on Instrumentation and Measurement*, 57(11):2608–2615, Nov. 2008.

[118] Renaud Santoro, Olivier Sentieys, and Sebastien Roy. On-the-fly evaluation of fpga-based true random number generator. In *Proceedings of the 2009 IEEE Computer Society Annual Symposium on VLSI*, ISVLSI '09, pages 55–60, Washington, DC, USA, 2009. IEEE Computer Society.

[119] Werner Schindler. Efficient online tests for true random number generators. In *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 103–117. Springer, 2001.

[120] Werner Schindler and Wolfgang Killmann. Evaluation criteria for true (physical) random number generators used in cryptographic applications. In *Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '02, pages 431–449, London, UK, UK, 2003. Springer-Verlag.

[121] J.M. Seguí and Universitat Oberta de Catalunya. Internet Interdisciplinary Institute (IN3). *Lightweight PRNG for Low-cost Passive RFID Security Improvement*. Universitat Oberta de Catalunya, 2011.

[122] Jian Shen, Dongmin Choi, S. Moh, and Ilyong Chung. A novel anonymous rfid authentication protocol providing strong privacy and security. In *Multimedia Information Networking and Security (MINES), 2010 International Conference on*, pages 584–588, Nov 2010.

[123] Ming.-Der Shieh, Jun.-Hong Chen, Wen.-Ching Lin, and Hao.-Hsuan Wu. A new algorithm for high-speed modular multiplication design. *IEEE Trans. on Circuits and Systems*, 56-I(9):2009–2019, 2009.

[124] S. Shrestha, M. Balachandran, M. Agarwal, V. V. Phoha, and K. Varahramyan. A chipless RFID sensor system for cyber centric monitoring applications. *IEEE Transactions on Microwave Theory and Techniques*, 57(5):1303–1309, May 2009.

[125] Kazem Sohraby, Mahmoud Daneshmand, Chonggang Wang, and Bo Li 0001. Performance analysis of rfid generation-2 protocol. *IEEE Transactions on Wireless Communications*, 8(5):2592–2601, 2009.

[126] S.S. Kumar and C. Paar. Are standards compliant elliptic curve cryptosystems feasible on rfid? Proc. Workshop RFID Security, 2006.

[127] B. Sunar, W. J. Martin, and D. R. Stinson. A provably secure true random number generator with built-in tolerance to active attacks. *IEEE Transactions on Computers*, 58:109–119, 2007.

[128] V. B. Suresh and W.P. Burleson. Entropy extraction in metastability-based trng. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 135–140, June 2010.

[129] C. Surman, R.A. Potyrailo, W.G. Morris, T. Wortley, M. Vincent, R. Diana, V. Pizzi, J. Carter, and G. Gach. Temperature-independent passive rfid pressure sensors for single-use bioprocess components. In *IEEE International Conference on RFID*, pages 78–84. IEEE Society, April 2011.

[130] I. E. Sutherland. Micropipelines. *Communications of the ACM*, 32(6):720–738, 1989.

[131] Chiu Chiang Tan, Bo Sheng, and Qun Li. Secure and serverless rfid authentication and search protocols. *IEEE Transactions on Wireless Communications*, 7(4):1400–1407, 2008.

[132] D. B. Thomas and W. Luk. The lut-sr family of uniform random number generators for fpga architectures. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 21(4):761–770, April 2013.

[133] Thomas E. Tkacik. A hardware random number generator. In *Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '02, pages 450–453, London, UK, UK, 2003. Springer-Verlag.

[134] UMC. Fsd0j_a umc 90 nm logic ll low-k process, 90 nm generic core 1d25v cell library. *Release Note 0.3, Rev. 2009Q2v2.0v2.0*, 2009.

[135] R. Vaidyanathaswami and A. Thangaraj. Robustness of physical layer security primitives against attacks on pseudorandom generators. *IEEE Transactions on Communications*, 62(3):1070–1079, March 2014.

[136] István Vajda and Levente Buttyán. Lightweight authentication protocols for low-cost rfid tags. In *2nd Workshop on Security in Ubiquitous Computing, in conjunction with Ubicomp 2003*, October 2003.

[137] B. Valtchanov, V. Fischer, and A. Aubert. Enhanced trng based on the coherent sampling. In *3rd International Conference on Signals, Circuits and Systems (SCS'09)*, pages 1–6, 2009.

[138] Vincent van der Leest, Erik van der Sluis, Geert-Jan Schrijen, Pim Tuyls, and Helena Handschuh. Cryptography and security. chapter Efficient Implementation of True Random Number Generator Based on SRAM PUFs, pages 300–318. Springer-Verlag, Berlin, Heidelberg, 2012.

[139] Ihor Vasyltsov, Eduard Hambardzumyan, Young-Sik Kim, and Bohdan Karpinskyy. Fast digital trng based on metastable ring oscillator. In *CHES*, pages 164–180, 2008.

[140] A. Vena, E. Perret, and S. Tedjini. Design of compact and auto-compensated single-layer chipless rfid tag. *Microwave Theory and Techniques, IEEE Transactions on*, 60(9):2913–2924, Sept 2012.

[141] John von Neumann. 13. various techniques used in connection with random digits. *Journal of Research of the National Bureau of Standards. Applied Mathematics Series*, 12(??):36–38, 1951.

[142] J. Walker. Randomness battery. 1998.

[143] Bin Wang and Maode Ma. A server independent authentication scheme for rfid systems. *IEEE Trans. Industrial Informatics*, 8(3):689–696, 2012.

[144] P.Z. Wieczorek. Dual-metastability fpga-based true random number generator. *Electronics Letters*, 49(12):744–745, 2013.

[145] Laurie Wiegler. Never lost in space: Nasa uses rfid on the space station. http://blog.atlasrfidstore.com/nasa-rfid-never-lost-space, 2014.

[146] A. Winstanley and M. Greenstreet. Temporal properties of self-timed rings. In *Correct Hardware Design and Verification Methods*, volume 2144 of *Lecture Notes in Computer Science*, pages 140–154. Springer Berlin Heidelberg, 2001.

[147] Knut Wold and Chik How Tan. Analysis and enhancement of random number generator in fpga based on oscillator rings. *Int. J. Reconfig. Comput.*, 2009:4:1–4:8, January 2009.

[148] Xilinx. Spartan-6 fpga data sheet:dc and switching characteristics, 2011.

[149] Sang-Kyung Yoo, Deniz Karakoyunlu, Berk Birand, and Berk Sunar. Improving the robustness of ring oscillator trngs. *TRETS*, 3(2):9, 2010.

[150] Y. z. Li, Y. b. Cho, N.-K. Um, and S.-H. Lee. Security and privacy on authentication protocol for low-cost RFID. In *Intl. Conf. on Computational Intelligence and Security*, volume 2, pages 1101–1104. IEEE Society, Nov. 2006.

[151] I. Zalbide, J. Vicario, and I. Velez. Power and energy optimization of the digital core of a Gen2 long range full passive RFID sensor tag. In *IEEE International Conference on RFID*, pages 125–133. IEEE Society, April 2008.

[152] Yuan Zhou, C.L. Law, and Jingjing Xia. Ultra low-power uwb-rfid system for precise location-aware applications. In *Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE*, pages 154–158, April 2012.

[153] Loic Zussa, Jean-Max Dutertre, Jessy Clédière, and Assia Tria. Power supply glitch induced faults on fpga: An in-depth analysis of the injection mechanism. In *IOLTS*, pages 110–115, 2013.